Titre: Title:	Recommandations conversationnelles dans le domaine des films
Auteur: Author:	Raymond Li
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Li, R. (2018). Recommandations conversationnelles dans le domaine des films [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3306/

UNIVERSITÉ DE MONTRÉAL

RECOMMANDATIONS CONVERSATIONNELLES DANS LE DOMAINE DES FILMS

RAYMOND LI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

RECOMMANDATIONS CONVERSATIONNELLES DANS LE DOMAINE DES FILMS

présenté par : LI Raymond
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. DESMARAIS Michel C., Ph. D., président
M. PAL Christopher J., Ph. D., membre et directeur de recherche
M. CHARLIN Laurent, Ph. D., membre et codirecteur de recherche
Mme ZOUAQ Amal, Ph. D., membre

REMERCIEMENTS

Je voudrais tout d'abord remercier le professeur Christopher Pal pour avoir supervisé ce projet de recherche. Lors de précieuses réunions, sa passion et son expérience ont su me guider avec sagesse tout au long de cette maîtrise. Je suis tout aussi reconnaissant envers mon co-directeur Laurent Charlin pour son regard critique et son discernement qui me poussait toujours à me questionner et à aller de l'avant. Je tiens à remercier également Hannes, Samira et Vincent pour leur participation régulière aux rencontres, et leur aide pour la récolte des données et l'implémentation. Mes remerciements vont aussi à mes camarades de laboratoire, et à toutes les autres personnes qui m'ont conseillé ou apporté de l'aide lors de cette maîtrise, notamment Louis-Henri, Sandeep, Adam, ainsi que les autres membres de l'équipe des éditeurs de Maluuba pour la vérification des conversations. Enfin, je remercie IBM d'avoir subventionné ma bourse de recherche, et Google pour leur participation financière pour la récolte des données.

RÉSUMÉ

Au delà des systèmes de recommandations basés sur les préférences passées des utilisateurs, une discussion avec un ami ou un libraire peut souvent ajouter de la richesse dans les recommandations obtenues, en plus d'être plus naturelle et agréable. Les recommandations conversationnelles sont un problème qui a attiré peu d'attention dans la recherche. Toutefois, les performances des nouvelles architectures de réseaux de neurones dans le domaine du dialogue permettent de s'attaquer à des problèmes aussi complexes que celui-ci. Une des raisons pour lesquelles ce problème n'a pas reçu beaucoup d'attention est le manque de données. Les jeux de données existant dans ce domaine sont souvent synthétiques ou très peu volumineux.

Au cours de ce projet de recherche, nous cherchons à créer un nouveau jeu de données pour les recommandations conversationnelles, et à l'exploiter pour développer un chatbot de recommandation. Nous choisissons le domaine cinématographique puisque c'est une application très classique des systèmes de recommandation.

Le nouveau jeu de données proposé comprend un ensemble d'entraînement de 10000 conversations, ainsi qu'un ensemble de test de 1300 conversations, dont les utilisateurs sont tous distincts de ceux de l'ensemble d'entraînement. Ces données ont été récoltées via Amazon Mechanical Turk. Dans chaque dialogue, un participant est censé demander des recommandations de films, tandis que l'autre doit les donner. Nous proposons une architecture de réseaux de neurones basée sur le Hierarchical Recurrent Encoder-Decoder utilisant ces données. Le jeu de données de 10000 dialogues est vraisemblablement très petit en comparaison d'autres corpus de dialogues utilisés en apprentissage profond, contenant jusqu'à plusieurs millions de dialogues. Ces données ne suffisent pas à entraîner un modèle comme le Hierarchical Recurrent Encoder-Decoder sans qu'il ne fasse du sur-apprentissage. Notre approche inclut donc deux sous-composantes qui peuvent être pré-entraînées, utilisant ainsi d'autres sources de données pour compenser la faible taille de notre jeu de données. Un premier module analyse le sentiment par rapport à chacun des films mentionnés dans la conversation. Ce sentiment servira d'entrée pour le module de recommandation. Ces deux modules sont entraînés en utilisant les étiquettes associées à chaque mention de film dans nos données, ainsi que le jeu de données MovieLens. Enfin, notre décodeur a une structure spéciale qui lui permet d'inclure ces recommandations explicites.

Nous expérimentons avec différentes variantes plus complexes de cette architecture, en introduisant une variable latente, ou des connexions supplémentaires. Il semble que la taille

restreinte du jeu de données ne nous permette pas d'utiliser des modèles trop complexes, et c'est notre premier modèle qui donne les meilleures résultats.

Une évaluation humaine montre cependant que notre modèle est significativement meilleur qu'un Hierarchical Recurrent Encoder-Decoder standard entraîné sur nos données, ce qui confirme la validité de notre approche. L'architecture proposée pourra ainsi servir de base de référence pour des travaux futurs sur ces données. Le jeu de données sera en effet rendu public, et rendra possible de nombreuses autres recherches sur les recommandations conversationnelles.

ABSTRACT

Beyond classical recommendation systems, a discussion with a friend or a librarian will often add richness in recommandations, and be more natural and enjoyable. Conversational recommendations have not drawn a lot of attention in research. But the recent advances in deep learning allow to tackle more and more complex problems. One reason why few works have dealt with conversational recommendations is the lack of data. Indeed, existing data sets of recommendation dialogues are often synthetic or too small.

During this research project, we gather a new data set of conversational recommendations, and use it to develop a recommendation chatbot. We chose to focus on movies, since it is a standard application of recommendation systems.

The newly gathered data set comprises a training set of 10000 conversations and a test set of 1300 conversations. The test set was collected with a distinct pool of workers. These dialogues were collected via Amazon Mechanical Turk. In each dialogue, one participant is the movie seeker and has to ask for recommendations, the other is the recommender. We propose a neural network architecture based on a Hierarchical Recurrent Encoder-Decoder and train it using our data set. 10000 dialogues is quite small in comparison to other corpora used in deep learning, that may contain up to several million conversations. 10000 conversations is not enough to train a vanilla Hierarchical Recurrent Encoder-Decoder without overfitting. We adress this issue by including sub-components that can be pre-trained using other data sources, thus compensating for the small size of the data set. A first module analyzes the sentiment with respect to each movie mentioned in the conversation. This sentiment will provide the input to the recommender module. These two modules are trained using the liked/disliked labels associated with each movie mention in our data set, and the MovieLens data set. We modify the decoder so it can make use of those explicit recommendations when generating sentences.

We also experiment with more complex variants of this architecure. We introduce a latent variable, or additional connections. It seems that the small size of the data doesn't allow us to correctly train such complex models, and our first model gives the best results.

However, a human evaluation shows that our model performs significantly better than a vanilla Hierarchical Recurrent Encoder-Decoder trained on our data, which confirms the validity of our approach. The proposed architecture will serve as a good baseline for future works on this data. Our data set will be publicly released, and will hopefully allow many other advances on conversational recommendations.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AMT	Amazon Mechanical Turk
CFG	Context Free Grammar
GRU	Gated Recurrent Unit
HIT	Human Intelligence Task
HRED	Hierarchical Recurrent Encoder-Decoder
LSTM	Long-Short Term Memory
MLP	MultiLayer Perceptron
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Network
SPARQL	SPARQL Protocol and RDF Query Language
VHRED	Variational Hierarchical Recurrent Encoder-Decoder

CHAPITRE 1 INTRODUCTION

Les systèmes de recommandations implantés dans des plate-formes comme Netflix ou Amazon se basent sur les objets achetés ou consultés pour recommander des produits similaires. Ce genre de système peut manquer de richesse dans les recommandations et sembler peu naturel. En revanche, une discussion avec un ami, un libraire ou un loueur de DVD offre un contexte plus libre qui permet d'exprimer avec des mots le genre de produit recherché. Une telle conversation peut être une expérience agréable, et mener à des idées de films à regarder. Le dialogue se présente donc comme une solution, parmi d'autres, permettant d'ajouter de la finesse à un système de recommandation, en plus d'apporter un cadre plaisant. Les systèmes de dialogue sont un sujet qui reçoit beaucoup d'attention, notamment par la communauté d'apprentissage profond. En effet, les méthodes utilisant les réseaux de neurones ont récemment prouvé qu'elles pouvaient produire des résultats impressionnants en traitement du langage. Les méthodes d'apprentissage profond nécessitent de grandes quantités de données. Malheureusement, les jeux de données existant en dialogue de recommandation sont trop petits, ou bien synthétiques. Dans cette maîtrise, nous allons donc répondre à cette demande de données. Nous expérimenterons avec les dernières architectures de réseaux de neurones sur ces nouvelles données, dans le but de développer un chat-bot de recommandation de films.

Avant de décrire le contenu de nos travaux, nous définissons les concepts de bases de l'apprentissage automatique, du traitement du langage et des systèmes de recommandation, nécessaires pour la bonne compréhension de ce mémoire.

1.1 Définitions et concepts de base

1.1.1 Apprentissage automatique

L'apprentissage automatique est l'étude des algorithmes qui sont capables d'adapter un modèle en fonction d'un jeu de données. En extrayant la structure des données, de tels algorithmes sont capables de résoudre des tâches qui pourraient être compliquées à coder autrement. Une grande partie des algorithmes d'apprentissage automatique, et notamment ceux de ce mémoire, utilisent des modèles paramétriques. Un modèle est une famille de fonctions, et le but de l'apprentissage est d'estimer les paramètres donnant la "meilleure" fonction de cette famille pour une tâche donnée. La signification du mot "meilleure" dans ce contexte sera clarifiée plus tard. Les algorithmes d'apprentissage automatique sont communément séparés en trois classes : l'apprentissage *supervisé*, *non supervisé*, et *par renforcement*. Nos travaux

concernent l'apprentissage supervisé, et nous en revoyons ici les principaux aspects. Voir Sutton and Barto (1998) pour un aperçu de l'apprentissage par renforcement, et Hastie et al. (2009) pour l'apprentissage non supervisé.

Apprentissage supervisé

Supposons qu'on observe le jeu de données $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$ qui sont des échantillons indépendants et identiquement distribués d'une variable aléatoire $(\mathbf{x}, y)$. L'apprentissage est dit supervisé car chaque $\mathbf{x}_i$ possède une étiquette y_i. L'apprentissage supervisé consiste à estimer une fonction $f : \mathbf{x} \mapsto y$ en utilisant un jeu de données $\mathcal{D}$. Lorsque y prend des valeurs discrètes, on parle de problème de *classification*. Par exemple, l'analyse de sentiment est un problème de classification : il s'agit de décider, pour une phrase donnée, si elle est positive ou négative (2 résultats possibles). Lorsque y prend des valeurs continues, on parle de problème de régression. Par exemple, la prédiction du prix d'une maison en fonction de son emplacement, de sa superficie, de sa date de rénovation est un problème de régression. Parmi les algorithmes d'apprentissage supervisé les plus performants et étudiés, on retrouve entre autres les Support Vector Machines (Cortes and Vapnik, 1995), les Random Forests (Breiman, 2001), ou encore les réseaux de neurones, décrits en section 1.1.2

Optimisation

Fonction de coût On mesure souvent la performance d'un modèle supervisé f avec une fonction de coût, ou fonction de perte, $l(y, f(\mathbf{x}))$. Par exemple, une perte quadratique s'écrit $l(y, f(\mathbf{x})) = \|y - f(\mathbf{x})\|^2$. Étant donné une fonction de perte l, on cherche à trouver la fonction f qui minimise le *risque* $R(f)$ défini par :

$$R(f) = \mathbb{E}_{(\mathbf{x},y)} \left[l(y, f(\mathbf{x})) \right] \tag{1.1}$$

La distribution des données étant inconnue, on minimise plutôt le *risque empirique* $\hat{R}(f, \mathcal{D})$:

$$\hat{R}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i)) \tag{1.2}$$

qui calcule la moyenne de l'erreur sur l'ensemble du jeu de données $\mathcal{D}$. Ainsi, si la fonction f est paramétrée par θ, alors on cherche

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{N} \sum_{i=1}^{N} l(y_i, f_\theta(\mathbf{x}_i))}_{\text{fonction de coût } J(\theta)} \tag{1.3}$$

Dans le cas d'un modèle probabiliste (non-supervisé), c'est-à-dire qu'on estime une densité de probabilité $P_\theta(\mathbf{x})$, on cherche plutôt à maximiser la vraisemblance :

$$\begin{aligned}\theta^* &= \arg\max_\theta P_\theta(\mathbf{x}_1, ..., \mathbf{x}_N) \\ &= \arg\min_\theta \underbrace{-\prod_{i=1}^{N} P_\theta(\mathbf{x}_i)}_{\text{fonction de coût } J(\theta)}\end{aligned} \tag{1.4}$$

Descente de gradient L'entraînement du modèle se résume donc à l'optimisation de la fonction de coût $J(\theta) \in \mathbb{R}$. Dans les équations 1.3 et 1.4, cette fonction dépend du jeu de données et peut avoir une expression très complexe. Il n'y a pas en général de solution analytique au problème d'optimisation $\arg\min_\theta J(\theta)$. Ainsi, la descente de gradient est une approche itérative qui permet de trouver des minima locaux d'une fonction différentiable. La fonction $J(\theta)$ définit une surface de coût, et l'idée est de prendre la direction qui, localement, donne la pente la plus "raide", pour descendre la surface de coût. Chaque mise à jour des paramètres du modèle s'écrit :

$$\theta \leftarrow \theta - \eta \nabla \mathbf{J}(\theta) \tag{1.5}$$

où $\eta \in \mathbb{R}$ est le *taux d'apprentissage* et $\nabla \mathbf{J}(\theta)$ le gradient de J par rapport à θ. Le taux d'apprentissage donne la taille du pas à effectuer dans la direction de la descente. Dans cette configuration, le gradient doit être calculé pour la fonction de perte J qui prend en compte tout le jeu de données. Pour accélérer le calcul et le taux de mise à jour des paramètres, il est courant de calculer le gradient seulement sur des *minibatchs*. Un minibatch contient quelques exemples d'entraînement. Plusieurs techniques ont pour but d'adapter le taux d'apprentissage au fil de l'entraînement, et d'utiliser une notion d'inertie pour accélérer l'apprentissage (Sutskever et al., 2013; Kingma and Ba, 2014).

Généralisation

La capacité d'un modèle désigne sa puissance représentative. Il n'y a pas une manière unique de mesurer la capacité d'un modèle, mais la dimension de Vapnik-Chervonenkis (Vapnik and Chervonenkis, 1968) en est un exemple. Un modèle à grande capacité possède typiquement beaucoup de paramètres, et sera capable d'approximer des fonctions très complexes. Les techniques d'optimisation nous permettent d'estimer les paramètres d'un modèle en utilisant des données d'entraînement, et des modèles à forte capacité peuvent atteindre une précision parfaite sur ces données simplement en les mémorisant. Le but de l'apprentissage automatique est toutefois qu'un modèle puisse *généraliser*, c'est-à-dire être performant sur de nouvelles

données, jamais vues pendant l'entraînement. Il est courant qu'un modèle trop complexe atteigne de très bonnes performances sur l'ensemble d'entraînement, mais pas sur de nouvelles données. On parle de *sur-apprentissage*. À l'inverse, un modèle pas assez complexe pour saisir toutes les variations dans les données souffre de *sous-apprentissage*.

Sur la figure 1.1, on entraîne des régressions polynomiales de différents degrés sur des données d'entraînement représentées par les points noirs. Les données de test, représentées par les triangles, ne sont pas utilisées lors de l'entraînement et permettent de comparer la performance des modèles sur de nouvelles données. Toutes les données (x_i, y_i) ont été générées pour suivre une parabole, avec un léger bruit :

$$y_i = \underbrace{10 - 0.1(x_i - 5)^2}_{\text{fonction à estimer}} + \epsilon_i \tag{1.6}$$

où les ϵ_i sont indépendants et identiquement distribués selon une loi normale centrée et d'écart-type 0.4. La régression de degré deux parvient donc à bien capter cette tendance. La régression linéaire est trop simple et ne permet pas de bien expliquer ces données, elle est en sous-apprentissage. La régression de degré 10 est encore plus proche des points d'entraînement que la régression de degré deux, mais on voit qu'elle donne de mauvaises prédictions sur les données de test : elle fait du sur-apprentissage.

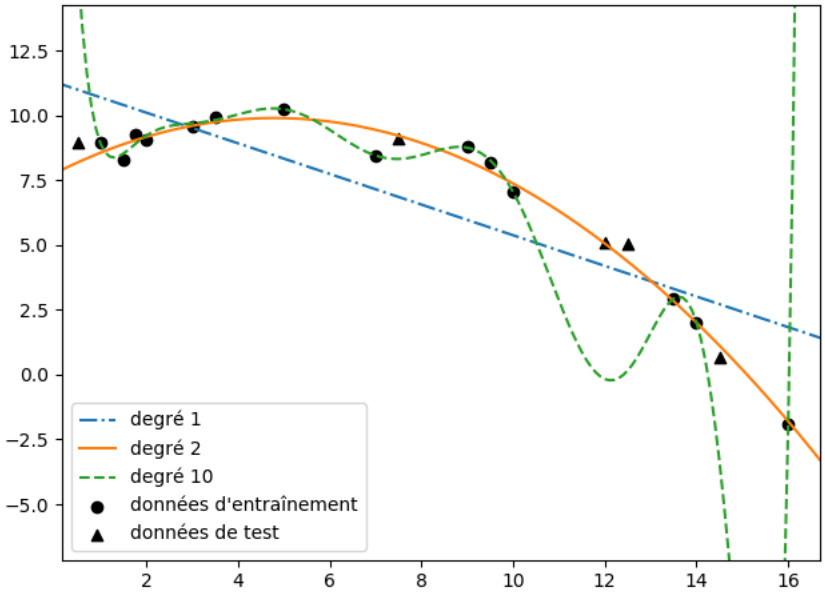


Figure 1.1 Illustration du sous et sur-apprentissage : Régressions polynomiales de différents degrés. Les données d'entraînement et de test ont été générées suivant la même fonction donnée en équation 1.6. Les données de test ne sont pas utilisées lors de l'entraînement.

Une méthode pour déterminer le bon modèle est de diviser le jeu de données en trois sous-ensembles : l'ensemble d'entraînement, l'ensemble de validation, et l'ensemble de test. L'ensemble d'entraînement est utilisé pour l'optimisation des paramètres du modèle, l'ensemble de validation pour les *hyper-paramètres* et stopper l'entraînement avant le sur-apprentissage, et l'ensemble de test pour rapporter les résultats expérimentaux. Les hyper-paramètres contrôlent la forme générale du modèle, et sa capacité. Il est important de les ajuster pour éviter le sur-apprentissage et le sous-apprentissage, pour obtenir une bonne généralisation. Le degré de la régression polynomiale, ou encore le taux d'apprentissage sont des exemples d'hyper-paramètres.

Plusieurs méthodes, dites méthodes de régularisation, permettent de limiter le sur-apprentissage. Il est courant d'ajouter un terme proportionnel à la norme des paramètres du modèle comme régularisation. Si $J(\theta)$ désigne la fonction de coût initiale, la nouvelle fonction de coût s'écrit :

$$\tilde{J}(\theta) = J(\theta) + \lambda\|\theta\| \tag{1.7}$$

où λ est un hyper-paramètre scalaire contrôlant l'importance de la régularisation, et $\|\cdot\|$ est une norme, typiquement la norme 1 ou la norme 2. L'*augmentation de données* consiste à générer de nouveaux exemples d'entraînement à partir des données existantes. Cette procédure est très utilisée en traitement de l'image par exemple (symétrie d'une image, rotation d'un petit angle, etc.).

1.1.2 Réseaux de neurones

Les réseaux de neurones sont inspirés des systèmes nerveux biologiques dans le sens où ces modèles, capables de modéliser des fonctions très complexes, sont composés d'une très grande quantité d'unités de calcul plus basiques, appelées neurones.

Soit l'entrée, ou *input*, $\mathbf{x} \in \mathbb{R}^d$, un neurone calcule une combinaison linéaire des composantes de $\mathbf{x}$, puis applique une *fonction d'activation* f au résultat obtenu :

$$y = f\left(\sum_{i=1}^{d} w_i x_i + b\right) = f(\mathbf{w}^T\mathbf{x} + b) \tag{1.8}$$

où $\mathbf{w}$ est le vecteur de poids, et b le biais du neurone. Lorsque la fonction d'activation est non linéaire, elle est l'élément clef qui permet au réseau de neurones d'apprendre des fonctions non-linéaires et de plus en plus complexes. Le choix des fonctions d'activation se fait entre autres sur les valeurs que la fonction peut prendre, sur les plages où la dérivée s'approche de zéro, ou encore la facilité avec laquelle un ordinateur peut calculer sa dérivée. Le tableau 1.1

donne des exemples de fonctions d'activation couramment utilisées.

Tableau 1.1 Fonctions d'activation fréquemment utilisées

Identité	$x \longmapsto x$
Sigmoïde (σ)	$x \longmapsto \frac{1}{1+e^{-x}}$
tanh	$x \longmapsto \frac{e^{x}+e^{-x}}{e^{x}+e^{-x}}$
Rectified Linear Unit (ReLU)	$x \longmapsto \max(0, x)$
Softplus	$x \longmapsto \log(1 + e^{x})$

Pour construire une couche de neurones, il suffit de combiner n neurones. Le résultat devient un vecteur dans $\mathbb{R}^n$:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1.9}$$

où $\mathbf{W} \in \mathbb{R}^{n \times d}$ est la matrice de poids, et $\mathbf{b} \in \mathbb{R}^n$ le vecteur de biais de la couche de neurones, et où f est appliquée composante par composante. Ici, les composantes de $\mathbf{W}$ et de $\mathbf{b}$ constituent les paramètres du modèle qui seront appris pendant l'entraînement. Le Multilayer Perceptron (MLP) consiste à appliquer successivement plusieurs couches de neurones. La figure 1.2 illustre l'architecture d'un MLP. Pour un MLP de trois couches par exemple, on a :

$$\mathbf{y} = f_3\left(\mathbf{W}_3 f_2(\mathbf{W}_2 f_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3\right) \tag{1.10}$$

L'équation 1.10 met en évidence qu'avec des fonctions d'activation linéaires, un réseau de neurone ne modélise que des fonctions linéaires. Toutefois, en utilisant des fonctions d'activation non-linéaires, un réseau de neurone de plusieurs couches est susceptible d'apprendre des fonctions très complexes. Le théorème d'approximation universel (Cybenko, 1989) indique qu'un MLP de deux couches (une couche cachée, et une couche de sortie) peut approximer n'importe quelle fonction continue, sur un ensemble compact (avec un nombre arbitrairement grand de neurones dans la couche cachée, et une certaine classe de fonctions d'activation).

1.2 Traitement du langage

Le traitement du langage, ou *Natural Language Processing (NLP)*, est l'interaction entre les ordinateurs et le langage humain et comprend un ensemble de techniques et algorithmes qui visent la compréhension du langage par les ordinateurs. Nous nous intéressons ici aux techniques basées sur l'apprentissage automatique. Un problème courant est la classification de phrase : l'analyse de sentiment (Pang et al., 2008), classifier une phrase objective ou subjective (Pang and Lee, 2004), la classification de questions (Li and Roth, 2002), etc. D'autres

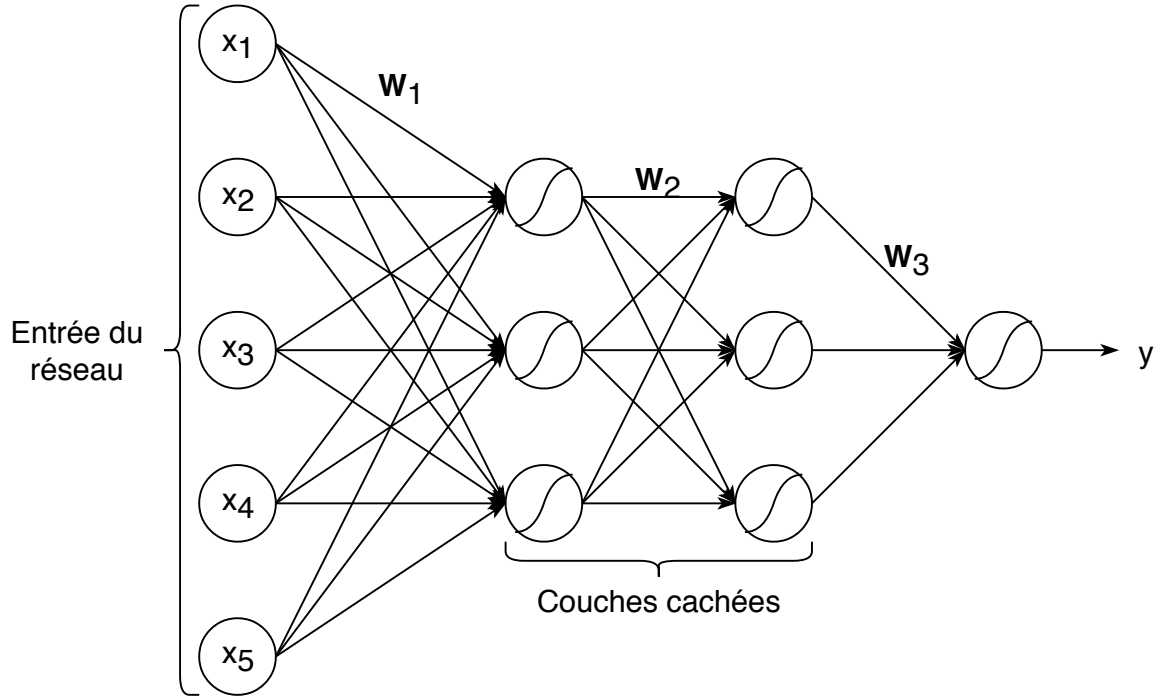


Figure 1.2 Exemple d'architecture d'un réseau de neurones à trois couches (dont deux couches cachées).

applications possibles peuvent nécessiter de la génération de texte comme la traduction, les tâches de question / réponse, ou le dialogue. Une phrase peut être vue comme une séquence de mots, de caractères, ou encore de segments de mots (Sennrich et al., 2015). Les approches basées sur les caractères ou les segments de mots ont l'avantage entre autres de pouvoir traiter des mots qu'ils n'ont pas observés dans l'ensemble d'entraînement. Elles seront aussi plus adaptés pour comprendre des langues utilisant des mots composés comme l'allemand. Toutefois nous nous intéressons ici à l'approche plus classique qui consiste à traiter les phrases comme des suites de mots.

Une phrase est une séquence $(w_1, ..., w_T)$ de mots où chaque mot w_i appartient à un vocabulaire V. Un marqueur spécial, <EOS> pour *End Of Sequence*, signifie la fin de la séquence. Ce marqueur fait partie du vocabulaire V et est ajouté à la fin de chaque phrase, de telle sorte qu'on a $w_T =$ <EOS>.

1.2.1 Réseaux de neurones récurrents

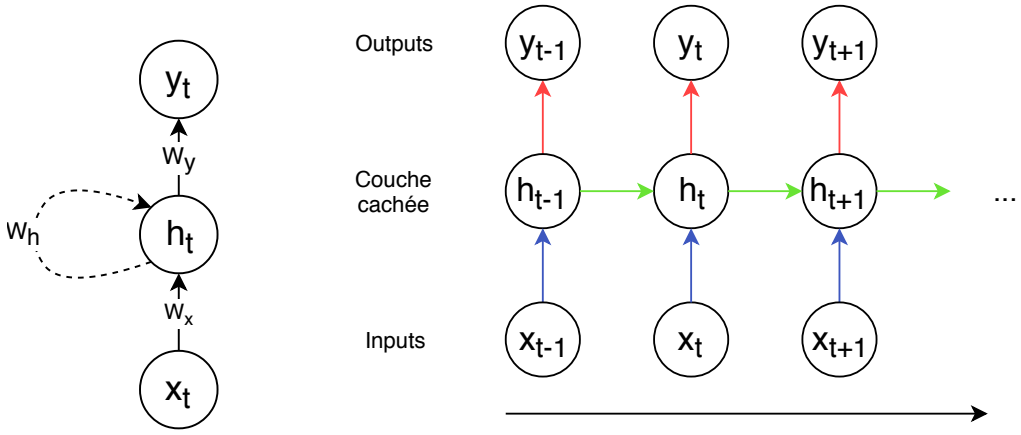
La taille des phrases varie et les MLP standards vus en section 1.1.2 ne peuvent traiter des données de taille variable. C'est pourquoi on utilise plutôt des Recurrent Neural Network (RNN). Les RNN sont une architecture de réseau de neurone qui permet de traiter des données

séquentielles, et donc potentiellement de taille variable. Soit la séquence d'entrée $\mathbf{x}_1, ..., \mathbf{x}_T$. Selon la tâche considérée, il peut y avoir des sorties $\mathbf{y}_t$ à chaque pas de temps t, ou bien seulement au dernier pas de temps T. Nous considérons par exemple le cas plus général où nous avons une séquence de sortie $\mathbf{y}_1, ..., \mathbf{y}_T$. Un RNN lit les entrées au fur et à mesure et maintient un état caché $\mathbf{h}_t$ de la manière suivante :

$$\mathbf{h}_t = \sigma_h \left(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}_h \right) \tag{1.11}$$

$$\mathbf{y}_t = \sigma_y \left(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y \right) \tag{1.12}$$

où $\mathbf{W}_x, \mathbf{W}_h, \mathbf{W}_y$ sont les matrices de poids, et $\mathbf{b}_h, \mathbf{b}_y$ les biais, qui seront appris pendant l'entraînement, et σ_h, σ_y les fonctions d'activation. Notons que les poids et biais du réseau ne dépendent pas du temps t, les mêmes poids sont utilisés à chaque pas de temps. La figure 1.3 illustre l'architecture des réseaux récurrents.



(a) Une itération de RNN

(b) RNN déployé sur plusieurs pas de temps. Les flèches de la même couleur partagent les mêmes matrices de poids.

Figure 1.3 Architecture d'un réseau récurrent.

L'entraînement des RNN pose un problème connu sous le nom de *vanishing* et *exploding gradient* (Hochreiter, 1991). Au fur et à mesure que l'on propage le gradient en remontant le temps, celui-ci a tendance à exploser, ou à tendre vers zéro (selon la taille des poids du réseau) à une vitesse exponentielle. Hochreiter and Schmidhuber (1997) introduit les Long-Short Term Memory (LSTM), architecture de neurone qui résout en partie ces problèmes. Le Gated Recurrent Unit (GRU) (Cho et al., 2014) est une autre architecture, illustrée figure 1.4, qui permet d'entraîner correctement des réseaux récurrents. Un GRU calcule des *reset gates* $\mathbf{r}_t$ qui déterminent quelle partie de l'état caché $\mathbf{h}_{t-1}$ sera utilisée pour calculer les *new gates* $\mathbf{n}_t$:

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r\right) \tag{1.13}$$

$$\mathbf{n}_t = \tanh\left(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{b}_{in} + \mathbf{r}_t \circ \left(\mathbf{W}_{hn}\mathbf{h}_{t-1} + \mathbf{b}_{hn}\right)\right) \tag{1.14}$$

où $\circ$ désigne la multiplication terme à terme et $\sigma : x \mapsto \frac{1}{1+e^{-x}}$ la fonction sigmoïde. Les *new gates* $\mathbf{n}_t$ représentent un candidat possible pour le nouvel état caché. Enfin, on utilise les *input gates* $\mathbf{z}_t$ pour combiner les *new gates* $\mathbf{n}_t$ avec l'ancien état caché $\mathbf{h}_{t-1}$:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z\right) \tag{1.15}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \circ n_t + \mathbf{z}_t \circ \mathbf{h}_{t-1} \tag{1.16}$$

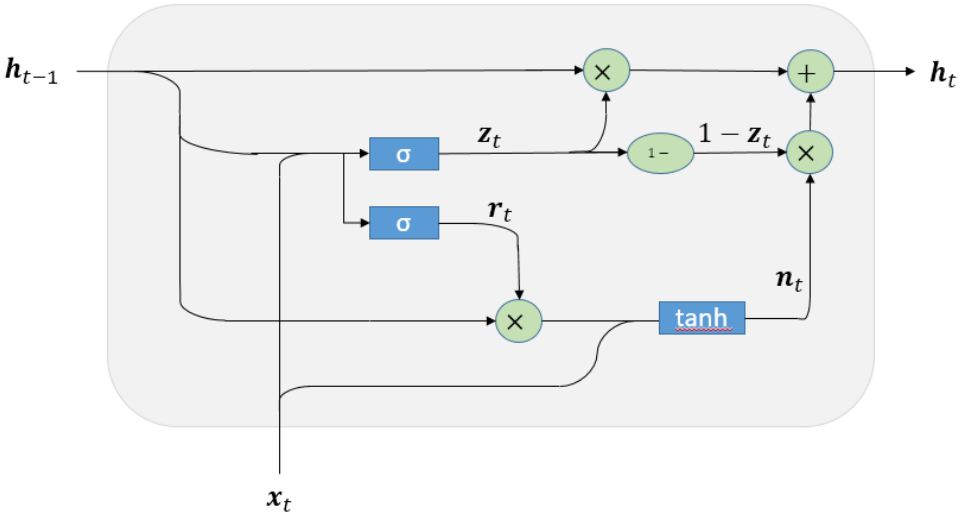


Figure 1.4 Architecture d'un GRU.

1.2.2 Réseaux de neurones récurrents bidirectionnels

Malgré les améliorations apportées par les LSTM (Hochreiter and Schmidhuber, 1997), une des principales limitations des réseaux récurrents est qu'ils peinent à apprendre les dépendances à long terme. Certains problèmes nécessitent également que la sortie à chaque pas de temps soit conditionnée, non seulement sur les mots qui précèdent, mais aussi ceux qui suivent. Les *RNN bidirectionnels* (Schuster and Paliwal, 1997) pallient ces deux problèmes en utilisant deux RNN distincts. L'un lit la séquence de gauche à droite, tandis que l'autre la lit de droite à gauche. Puis, la couche suivante, ou bien la couche de sortie, utilise la concaténation des états cachés des deux RNN comme état caché global. La figure 1.5 illustre cette architecture.

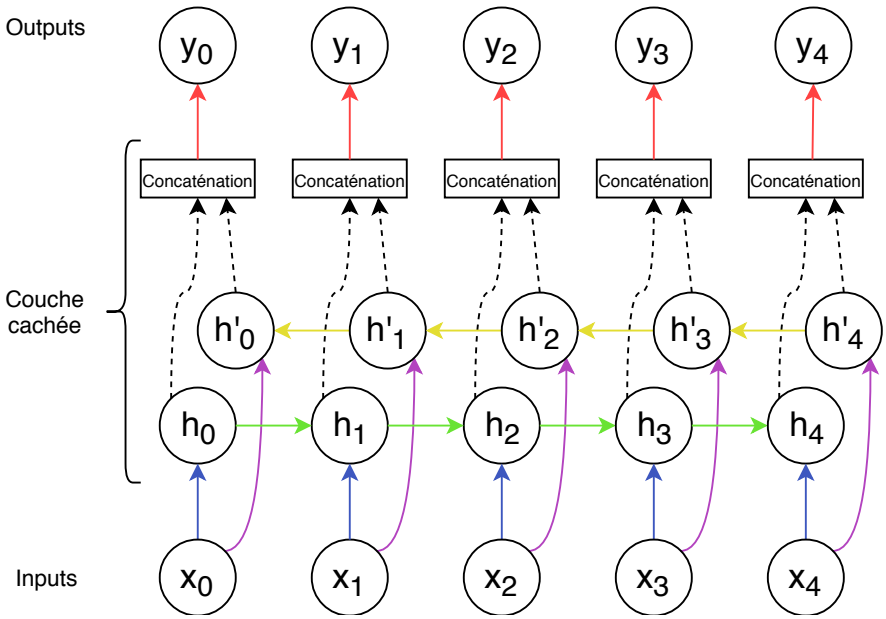


Figure 1.5 RNN bidirectionnel déployé sur une séquence de 5 éléments. Les flèches de la même couleur partagent les mêmes matrices de poids.

1.2.3 Plongement de mots

Il est habituel de représenter chaque mot du vocabulaire par un vecteur *one-hot*. Une telle suite de vecteurs one-hot donnerait des entrées très peu denses, et de grande dimension (de la taille du vocabulaire V). C'est pourquoi on utilise couramment des *word embeddings*, ou plongements de mots (Hinton et al., 1986). L'idée est d'associer à chaque mot un vecteur de *features* continu de $\mathbb{R}^d$. Soit la suite de vecteurs *one-hot* $(w_1, ..., w_T)$ et la matrice de plongement $A \in \mathbb{R}^{d \times |V|}$ où d est la taille des vecteurs de plongement. La séquence de vecteurs one-hot devient une séquence de vecteurs dans $\mathbb{R}^d$:

$$(Aw_1, ..., Aw_T) \tag{1.17}$$

qui peut être donnée en entrée au RNN. Les vecteurs de plongement sont censés capturer la sémantique des mots. Ainsi, deux mots de sens proches auront des représentations similaires : les mots "planète" et "planètes" devraient avoir des représentations très proches. On observe même parfois une "arithmétique" de mots très intéressante, comme par exemple : $vecteur(\text{"}Roi\text{"}) - vecteur(\text{"}Homme\text{"}) + vecteur(\text{"}Femme\text{"}) = vecteur(\text{"}Reine\text{"})$ (Mikolov et al., 2013a).

On utilise souvent des plongements de mots pré-entraînés sur un corpus à part. Plusieurs approches ont été explorées pour apprendre des plongements de mots. Skip-gram utilise chaque mot pour prédire son contexte, tandis que Continuous Bag-of-words utilise le contexte

d'un mot pour prédire ce mot (Mikolov et al., 2013b). Pennington et al. (2014) se base sur la matrice de co-occurrence des mots sur un très large corpus. Bojanowski et al. (2016) représente les mots comme des ensembles de n-grammes, ce qui permet d'obtenir des représentations même pour des mots non-observés dans l'ensemble d'entraînement.

1.2.4 *Teacher forcing*

Les RNN peuvent aussi être utilisés comme modèles probabilistes pour modéliser le langage en utilisant la factorisation suivante :

$$p(w_1, ..., w_T) = \prod_{t=1}^{T} p(w_t | w_1, ..., w_{t-1}) \tag{1.18}$$

À chaque pas de temps, la sortie du modèle est une distribution sur l'ensemble du vocabulaire qui prédit le prochain mot. En *teacher forcing* (Williams and Zipser, 1989), le but est alors de maximiser la log-vraisemblance de la séquence $w_1, ..., w_T$. À chaque pas de temps, le modèle prend en entrée le mot précédent w_{t-1} et est entraîné à prédire w_t.

1.2.5 Génération de phrases

Au moment de la génération, ou inférence, trouver la séquence de plus grande probabilité est impossible en pratique puisque le nombre de séquences possibles grandit exponentiellement avec la taille des séquences. Une solution, parfois appelée *free running*, est de faire avancer le modèle pas à pas, et de générer les mots un à un. Chaque mot qui est généré est ensuite réinséré comme entrée au modèle au prochain pas de temps. Lorsque le marqueur <EOS> est généré, la génération s'arrête. La figure 1.6 illustre le procédé de génération de texte.

L'algorithme de *recherche en faisceau*, ou *beam search*, est une heuristique employée en parcours de graphe où un certain nombre d'hypothèses sont évaluées en parallèle, pour écarter les phrases peu prometteuses (Bisiani, 1992). C'est une alternative moins gloutonne que le *free running* (qui correspond au cas où la largeur du faisceau vaut 1). Cette méthode de recherche permet de générer des phrases dont la probabilité selon le modèle est plus élevée, et donne en général des générations de meilleure qualité.

L'entraînement et l'inférence sont différents dans le sens où pour prédire w_t l'entraînement utilise le vrai mot w_{t-1}, tandis que l'inférence utilise l'approximation générée par le modèle $\hat{w}_{t-1}$. Cette différence fait que le modèle se trouve, lors de l'inférence, potentiellement dans des états qui n'ont jamais été explorés lors de l'entraînement. Ce phénomène est appelé *biais d'exposition*, ou *exposure bias*, référant au fait que lors de l'apprentissage le modèle est

exposé à la distribution des données d'entraînement et non à ses propres générations. Il est possible d'entraîner le modèle dans le même mode que la génération, en utilisant les mots générés par le modèle comme entrées. La moindre erreur va toutefois pénaliser le modèle et le mener dans une mauvaise direction. Cette procédure d'entraînement converge extrêmement lentement, voire pas du tout. Bengio et al. (2015) propose de commencer l'entraînement en *teacher forcing*, puis de petit à petit incorporer de plus en plus les générations du modèle dans l'entraînement. Une telle approche semble donner de meilleurs résultats sans pour autant allonger l'entraînement.

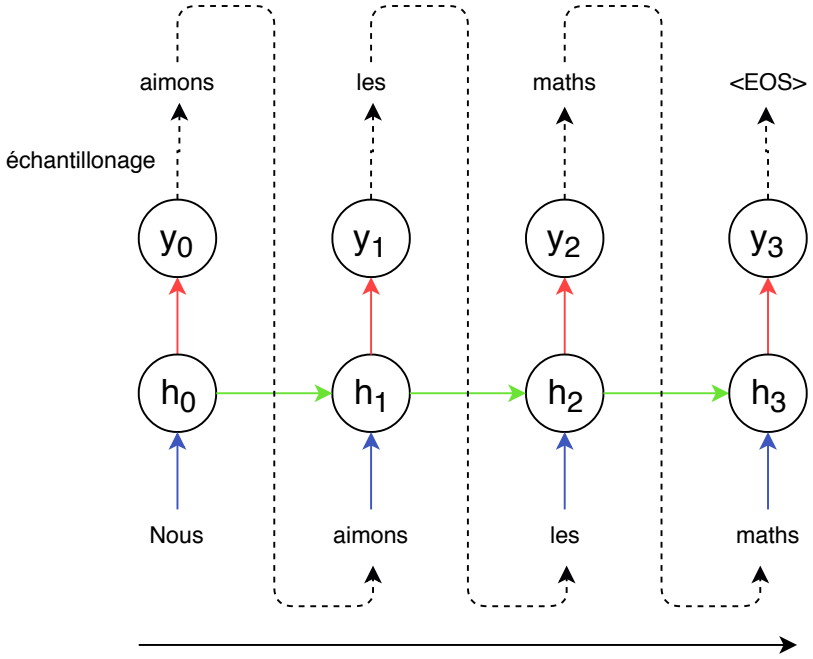


Figure 1.6 Génération de phrases avec un RNN. On donne dans cet exemple le mot "Nous" comme premier mot de la phrase. Chaque mot généré par le RNN est donné comme entrée au prochain pas de temps, jusqu'à ce que le réseau génère le marqueur <EOS>.

1.3 Systèmes de recommandation

Dans cette section, nous voyons quelques concepts importants abordés dans les systèmes de recommandation. On dispose d'une liste d'objets (des livres, des films, des documents, etc.) et d'une liste d'utilisateurs. Un système de recommandation est un modèle qui prédit si un utilisateur est susceptible d'aimer un objet. On distingue en général deux catégories de systèmes de recommandation : les approches basées sur le contenu, et le *filtrage collaboratif*.

L'approche basée sur le contenu consiste à établir des profils d'objets à partir de leurs attributs

(pour un film par exemple, son genre, des tags, sa distribution, etc.) et d'utiliser ces profils pour faire des recommandations à un utilisateur. Le *Music Genome Project*[1] est un exemple de ce genre de systèmes de recommandation. Un modèle est entraîné à déterminer un profil pour chaque chanson, basé sur des centaines de caractéristiques musicales pré-définies. Ce profil résume l'identité musicale du titre, et l'information qui permet de déterminer si un utilisateur donné l'aimerait ou pas.

Le filtrage collaboratif repose également sur la comparaison de profils d'objets et d'utilisateurs, mais ces profils résultent de l'analyse du comportement des utilisateurs et des dépendances entre les objets. Nous en montrons ici quelques aspects. Su and Khoshgoftaar (2009) fait un plus large résumé des techniques de filtrage collaboratif.

1.3.1 Factorisation de matrice

La *factorisation de matrice* est un modèle courant pour calculer les profils (Koren et al., 2009). Soit m le nombre d'utilisateurs, n le nombre d'objets et $\mathbf{R} \in \mathbb{R}^{m \times n}$ la matrice des préférences, ou *ratings*, utilisateur-objet partiellement observée. L'objectif est d'approximer $\mathbf{R}$ avec une matrice de la forme $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T$, où $\mathbf{P} \in \mathbb{R}^{m \times d}$, $\mathbf{Q} \in \mathbb{R}^{n \times d}$, et d est la dimension des profils. On note $\mathbf{M}_i$ la i-ième ligne de la matrice $\mathbf{M}$, et M_{ij} sa composante sur la i-ième ligne et la j-ième colonne. L'interaction entre l'utilisateur u et l'objet i s'écrit alors :

$$\hat{R}_{ui} = \mathbf{P}_u \mathbf{Q}_i^T \tag{1.19}$$

On obtient les matrices de profil en résolvant le problème d'optimisation suivant :

$$\min_{\Theta} \sum_{u,i} \left(\mathbf{R}_{ui} - \hat{\mathbf{R}}_{ui} \right)^2 + \underbrace{\lambda \|\Theta\|^2}_{\text{régularisation}} \tag{1.20}$$

où Θ est constitué des deux matrices $\mathbf{P}$, $\mathbf{Q}$, et la somme se fait sur tous les *ratings* observés dans l'ensemble d'entraînement. Pour également prendre en compte les tendances individuelles des utilisateurs et des objets, on ajoute des biais :

$$\hat{r}_{ui} = \underbrace{\mu + \mathbf{b}_u + \mathbf{b'}_i}_{\text{biais}} + \underbrace{\mathbf{P}_u \mathbf{Q}_i^T}_{\text{interaction utilisateur-objet}} \tag{1.21}$$

où μ, $\mathbf{b}$ et $\mathbf{b}'$ sont les biais global, d'utilisateur, et d'objet, également appris avec les matrices $\mathbf{P}$ et $\mathbf{Q}$. Une principale limitation de cette approche, le *cold start problem*, est qu'elle est incapable de faire des prédictions pour des utilisateurs ou des objets jamais observés dans

1. `https://www.pandora.com/about/mgp`

l'ensemble d'entraînement. En effet, les prédictions reposent sur des représentations propres à chaque objet et utilisateur stockées dans les matrices $\mathbf{P}$ et $\mathbf{Q}$, apprises pendant l'entraînement. Une approche utilisant le contenu des objets n'a pas ce problème.

1.3.2 Auto-encodeur pour les systèmes de recommandation

Les auto-encodeurs sont des réseaux de neurones dont l'apprentissage n'est pas supervisé, introduits par Hinton and Salakhutdinov (2006) pour la réduction de dimensionnalité. Le modèle est constitué d'un *encodeur* et d'un *décodeur*. L'*encodeur* transforme l'entrée $\mathbf{x}$ en un code $\mathbf{h}$, et le *décodeur* produit la reconstruction $\tilde{\mathbf{x}}$. L'objectif d'entraînement est de minimiser l'erreur de reconstruction, par exemple la distance entre $\mathbf{x}$ et sa reconstruction $\tilde{\mathbf{x}}$. Ainsi, si la dimension du code est plus faible que celle de l'entrée, on crée un goulot d'étranglement qui force le réseau à compresser l'information.

Vincent et al. (2008) propose d'ajouter du bruit aux entrées lors de l'entraînement pour rendre le modèle plus robuste. L'objectif du *denoising auto-encoder* est de reconstruire l'entrée sans bruit. Le modèle apprend ainsi à débruiter les données.

Sedhain et al. (2015) utilise un auto-encodeur pour fournir des recommandations. Nous revoyons ce système en détails section 3.4.3.

1.3.3 Recommandations et traitement du langage

Plusieurs travaux utilisent le langage pour améliorer les systèmes de recommandation. En plus de prédire les *ratings*, Almahairi et al. (2015) modélise les critiques des utilisateurs à partir des profils d'objets. Cet apprentissage multi-tâche permet de régulariser les profils appris et d'améliorer les recommandations.

Certaines approches combinent le filtrage collaboratif avec un contenu textuel associé aux objets (un résumé, un synopsis, une description, etc.) qui donne des informations plus précises sur les objets, et peut notamment être utilisé en *cold start problem* (Wang and Blei, 2011; Gopalan et al., 2014; Bansal et al., 2016). Par exemple, Bansal et al. (2016) représente un objet i, décrit par un texte X_i, par $\mathbf{Q}_i + g(X_i)$ à la place de $\mathbf{Q}_i$ dans l'équation 1.21. La fonction g peut typiquement être modélisée par un réseau récurrent. Ainsi, le contenu X_i permet d'obtenir une représentation d'un objet même s'il n'était pas présent dans l'ensemble d'entraînement.

1.4 Éléments de la problématique

Les systèmes de recommandation et notamment le filtrage collaboratif sont très bien établis de nos jours. De grandes compagnies comme Amazon ou Netflix les utilisent couramment pour fournir des recommandations personnalisées, en fonction des différents objets examinés ou achetés par les utilisateurs. Toutefois de telles recommandations ne peuvent pas s'adapter au désir immédiat de l'utilisateur. Par exemple, ce modèle ne permet pas de répondre à un utilisateur qui cherche un objet différent de ce qu'il a l'habitude de consulter. Nous faisons le choix de nous concentrer sur le domaine cinématographique, puisque c'est une des applications les plus courantes des systèmes de recommandation. Mettons que je sois en général un amateur de films de science fiction, mais aujourd'hui avec des amis nous avons décidé de regarder un film d'horreur, mais pas un film qui ne vous fait que sursauter, nous voulons un aspect psychologique en plus. Ou alors je désire regarder un bon film de science fiction avec Johnny Depp, parce que c'est mon acteur préféré. Les modèles de recommandation classiques manquent souvent de richesse et ne sauraient quoi me conseiller dans ce cas. Dans ce genre de situation, un libraire ou un loueur de DVD saurait très bien me conseiller simplement en dialoguant avec moi, pour comprendre quel genre de recommandation je recherche. En effet on a parfois des requêtes qu'on exprime plus naturellement par le langage. Le dialogue permettrait à l'utilisateur d'utiliser les mots pour définir ses préférences et pourrait apporter plus de nuances aux recommandations. Nous nous plaçons dans ce genre de contexte et voulons développer un système capable, par le dialogue, de comprendre les goûts cinématographiques de l'utilisateur, et de recommander des films en conséquence. Nous appellerons cette tâche les *recommandations conversationnelles*.

Les réseaux de neurones suscitent de plus en plus d'attention, notamment pour leur capacité à modéliser des dialogues. On distingue souvent dans la littérature les dialogues avec objectif (réservation d'un billet de train ou support technique par exemple) du *bavardage*, ou *chit-chat* en anglais (dialogue plus libre, dont le but est de divertir). Les recommandations conversationnelles s'apparentent de prime abord aux dialogues avec objectif. Toutefois, dans le cas des recommandations, il y a rarement une seule bonne réponse, ce qui rend l'objectif difficile à définir. De plus le seul retour de l'utilisateur à propos d'une recommandation faite pendant un dialogue est sa réponse. On sait potentiellement si la personne trouve qu'il s'agit d'une bonne recommandation, ce qui est bien plus difficile à quantifier qu'un *rating* d'une personne qui a vu le film. On observe par ailleurs que, par rapport aux dialogues avec objectif habituels, les recommandations conversationnelles se rapprochent souvent d'un dialogue plus libre et naturel.

Les recommandations conversationnelles sont un problème assez nouveau et qui n'a pas

reçu beaucoup d'attention dans la littérature. Cette tâche laisse la place à de nombreuses contributions. En outre, le seul jeu de données de dialogues de recommandation de films conséquent est, à notre connaissance, le Facebook Movie Dialog Data Set (Dodge et al., 2015). Or ce dernier est un jeu de données synthétique, généré à partir de phrases types, et n'est probablement pas suffisant pour entraîner un modèle de recommandation conversationnelle. Il est crucial de créer un jeu de données de dialogues de recommandations de film, en langage naturel. En plus de nous permettre d'expérimenter de nouvelles architectures, ce jeu de données pourrait susciter l'intérêt de la communauté pour cette nouvelle tâche. Il est important de réfléchir d'une part à ce que ce jeu de données doit prendre en compte pour nous permettre d'entraîner un modèle, et d'autre part à ce qu'il propose du nouveau pour la communauté. Il faut par ailleurs garder à l'esprit que ces données doivent être récoltées auprès d'êtres humains, dans le but d'obtenir des dialogues aussi naturels que possible.

En ce qui concerne le système de dialogue, nous nous orientons vers des modèles basés sur des réseaux de neurones récurrents, puisque c'est une architecture qui a prouvé qu'elle pouvait produire des résultats convaincants sur d'autres tâches de dialogue (Sordoni et al., 2015; Weston, 2016; Das et al., 2017; Zhang et al., 2018). Tout d'abord, on sait que l'utilisation de nombreux mots avec les réseaux récurrents est un défi très étudié dans la littérature. En effet, la taille du vocabulaire représente la dimension des données en entrée et en sortie du réseau. Ainsi, l'utilisation de grands vocabulaires requiert beaucoup de mémoire. De plus, les données en sont d'autant plus éparses, ou *sparse*, ce qui complique l'apprentissage des modèles. Un modèle de dialogue appliqué au domaine cinématographique sera inévitablement amené à manipuler de nombreuses entités, des noms de films et d'acteurs notamment, ce qui constitue un challenge en soi. Une deuxième difficulté est d'orienter un tel système dans le but spécifique de faire des recommandations. Le principe d'un système de recommandation est que l'algorithme doit acquérir une connaissance des objets et des utilisateurs pour prédire ce que l'utilisateur est susceptible d'aimer. Nous pensons que le modèle pourra grandement s'améliorer en extrayant des connaissances d'autres jeux de données sur les films : des graphes de connaissances, des bases de données (donnant de l'information sur les meta-données des films) ou bien des données de *ratings* (qui nous permettraient de comprendre les dépendances et les similarités entre tous ces films). On disposerait alors de plusieurs types de données : notre jeu de données, i.e. des dialogues de recommandation, et des données sur les films. Il s'agit de réfléchir à comment intégrer plusieurs types de données en un seul modèle, pour faire des recommandations pertinentes à travers un dialogue avec l'utilisateur.

1.5 Objectifs de recherche

L'objectif de recherche principal est d'explorer des modèles utilisant des réseaux de neurones pour voir dans quelle mesure de tels modèles sont capables d'appréhender des dialogues dans le domaine précis des recommandations de films.

En découlent plusieurs objectifs plus spécifiques. Tout d'abord, nous explorerons les différents travaux sur les systèmes de dialogue et de recommandation, pour comprendre les enjeux des recommandations conversationnelles. Puis, nous voulons récolter un nouveau jeu de données spécifiquement pour cette tâche, et dans le but de le rendre public. Cette récolte se fera par le biais de la plateforme Amazon Mechanical Turk [2]. En utilisant ce jeu de données, nous développerons et entraînerons un modèle capable de faire des recommandations de films à travers un dialogue. Notre objectif est de tester différentes architectures, et d'évaluer laquelle donne de meilleurs résultats sur notre jeu de données. Nous avons aussi pour objectif d'incorporer au modèle des connaissances extérieures sur les films. Ces connaissances peuvent se présenter sous plusieurs formes : un graphe de connaissance sur le domaine cinématographique, ou encore des *ratings*.

1.6 Plan du mémoire

Ce mémoire commence par une revue critique de la littérature au chapitre 2. Cette revue permet de mieux comprendre les travaux liés au problème des recommandations conversationnelles, les enjeux, et les difficultés que l'on peut attendre. Le chapitre 3 est un article que nous avons soumis à la conférence *Neural Information Processing Systems*. Cet article présente la partie principale de nos travaux, notamment la récolte de données, ainsi qu'un modèle combinant dialogue et recommandations. Puis, dans le chapitre 4, nous complétons la partie précédente en apportant des commentaires sur le contenu de l'article. Nous présentons aussi d'autres variantes du modèle proposé, qui viennent éclairer les travaux effectués et donner des pistes pour poursuivre cette recherche. Enfin, nous concluons en faisant une synthèse des travaux liés à cette maîtrise, mettons en évidence les limitations de notre approche, et proposons des améliorations possibles.

2. `https://www.mturk.com/`

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous revoyons les travaux utiles à la compréhension du chapitre 3 de ce mémoire. Nous examinons plus en détails les autres jeux de données combinant dialogue et recommandation, et donnons un aperçu un peu plus large des autres travaux pertinents qui ont été faits sur les agents de dialogue.

2.1 Systèmes de dialogue

Certains travaux voient le dialogue comme un échange multimodal de sons, de gestes, d'expressions faciales (Cassell, 2000; Wahlster, 2006). Dans le cadre d'un échange parlé, un système de dialogue doit comprendre un module de reconnaissance de la parole, pour traduire le son en texte, un module intermédiaire qui cherche à comprendre l'interlocuteur puis à générer une réponse, et enfin un synthétiseur qui transforme le texte en parole. Nous nous intéressons dans ce mémoire aux travaux sur les agents de dialogue écrit. Nous voyons le dialogue comme un échange de messages textuels et laissons de côté la partie de reconnaissance vocale et les autres modes de communication.

Les premiers travaux sur les agents de dialogue se basent principalement sur des états et des règles définis manuellement par des humains. Avec son système ELIZA, Weizenbaum (1966) détecte certains mots-clef dans la phrase de l'utilisateur, puis génère une réponse en suivant certaines règles écrites à la main. L'agent de dialogue bien connu A.L.I.C.E. (Wallace, 2009) est également basé sur une liste de règles. Les systèmes de dialogue ont été utilisés commercialement pour les services à la clientèle téléphoniques, comme pour obtenir des horaires de trains (Aust et al., 1995), ou bien réserver des billets d'avion (Bobrow et al., 1977). Bobrow et al. (1977) utilise une liste de *champs*, qui doivent être remplis pour comprendre le besoin de l'utilisateur. On parle de *slot filling*. Ces champs sont pré-définis par un expert, et sont remplis petit à petit par le modèle. Les champs peuvent être par exemple le lieu de départ, d'arrivée, et la date du voyage. Le système Phoenix (Ward and Issar, 1994) utilise des grammaires non-contextuelles de plusieurs milliers de règles définies par un expert pour analyser les phrases de l'utilisateur et remplir ces champs. Les grammaires non-contextuelles, ou *Context Free Grammar (CFG)* en anglais, ont longtemps été utilisées pour modéliser le langage (Chomsky, 1956). Formellement, une grammaire non-contextuelle est composée d'un ensemble V de variables, d'un ensemble A de symboles terminaux (disjoint de V), d'un axiome $S \in V$, et d'un ensemble $\mathcal{P}$ de règles de production. Une règle de production est de la forme $X \mapsto \alpha$ où X est une variable et α une séquence de variables et de symboles terminaux. Cet

ensemble de règles définit ainsi les séquences de symboles, ou phrases, qui sont correctes selon cette grammaire.

Il est probable que nous n'arrivions jamais à énumérer toutes les règles qui interviennent dans notre processus de compréhension, réflexion, puis énonciation d'une réponse au cours d'un dialogue, au vu de la complexité du langage humain. C'est pourquoi, dans les travaux plus récents, des approches utilisant l'apprentissage automatique ont été explorées.

De nombreux travaux tendent à faire la distinction entre deux catégories de systèmes de dialogue : les systèmes avec but, ou *goal-driven*, comme des services de réservation ou de support technique, et sans but, *non-goal-driven*, tels que les chatbots, censés converser sans avoir un domaine particulier. Même si la distinction est assez floue, puisque tout système de dialogue a bien un objectif *in fine*, les systèmes avec but ont souvent une mesure de performance bien définie. Dans le cas d'un système de réservation de billet d'avion par exemple, le fait que la personne ait réservé un billet ou non à la fin de la conversation donne la mesure de performance. Les recommandations conversationnelles ont clairement pour but de donner des recommandations qui satisfont l'utilisateur. Toutefois il n'existe pas de mesure de performance explicite. Notre approche n'est pas clairement dans l'une ou l'autre des catégories, et tire son inspiration des deux. En effet, notre modèle est basé sur une architecture hiérarchique (Sordoni et al., 2015) élaborée pour les dialogues sans but, et incorpore également des modules intermédiaires comme le font souvent les systèmes avec but. Chen et al. (2017) fait une revue des récentes applications de l'apprentissage statistique aux systèmes de dialogue. Nous revoyons ici les principaux travaux effectués, dont notre approche s'inspire.

2.1.1 Agents de dialogue avec but

Un agent de dialogue avec but comprend typiquement quatre composantes :

- Un interpréteur de langage, ou *Natural Language Understanding*. Ce module prédit, pour chaque message reçu, l'intention de l'utilisateur, sous forme de labels pré-définis. Il s'agit donc d'un classifieur dont l'entraînement requiert des paires de message et d'intention de l'utilisateur. Cela nécessite donc de demander aux humains d'annoter chaque message avec l'intention correspondante.
- Un traqueur d'état de dialogue, ou *Dialogue State Tracker*, qui estime à chaque étape de la conversation l'état du dialogue. L'état du dialogue est typiquement un ensemble de valeurs associées à des *champs*. Une liste de champs pourrait être : le lieu de départ, d'arrivée, et la date du voyage. Ici encore, ce classifieur requiert des données annotées pour être entraîné.
- Le sélectionneur de réponse choisit, en fonction de l'état du dialogue, la réponse ou

l'action à effectuer. Par exemple, l'agent peut choisir de fournir des informations (comme l'horaire du prochain train), ou bien de poser une question (comme demander de répéter la ville de départ).

— Le générateur de langage, ou *Natural Language Generation* génère la phrase qui sera envoyée à l'utilisateur, à partir de la réponse choisie. Ce module utilise souvent des règles définies à la main, comme par exemple : "Votre train partira à la gare X à Y" où la gare X et l'heure de départ Y sont définies dans la réponse du modèle.

2.1.2 Systèmes de dialogue sans but et apprentissage profond

Les systèmes de dialogue sans but sont destinés à discuter de sujets plus ouverts, et à être divertissants plus qu'utiles. Les réseaux de neurones génératifs et les méthodes d'extraction sont les deux principales approches aux chatbots sans but.

Réseaux de neurones génératifs

Des systèmes de dialogue de bout-en-bout, ou *end-to-end systems* en anglais, basés sur des réseaux de neurones sont de plus en plus populaires. Un avantage de ces méthodes est qu'elles ne nécessitent pas d'annotation intermédiaire pour être entraînées. Contrairement aux approches qui utilisent des règles définies à la main pour créer la réponse du modèle, les réseaux de neurones sont capables de générer de toutes nouvelles phrases dans un nouveau contexte. Comme vu en section 1.2.5, les réseaux de neurones peuvent être utilisés comme modèles génératifs. Ritter et al. (2011) voit le problème de réponse à des Tweets comme un problème de traduction. Ils utilisent un modèle *séquence-à-séquence*. Le principe est d'encoder la phrase en input avec un RNN. Puis un second RNN décode le code obtenu pour générer la phrase en output, comme illustré sur la figure 2.1. Bahdanau et al. (2014) ajoute un mécanisme d'attention qui permet au modèle de se focaliser sur différentes parties de l'input au cours du décodage.

Pour un agent de dialogue, il est crucial de pouvoir également prendre en compte les échanges passés en plus du dernier message reçu. Sordoni et al. (2015) propose d'adapter le modèle séquence-à-séquence en utilisant un encodeur hiérarchique, décrit en section 3.4.1 et illustré à la figure 2.2. Xing et al. (2017) utilise en plus de cela un système d'attention hiérarchique pour focaliser sur les parties importantes au sein et parmi les messages précédents.

Un des principaux défauts des réseaux génératifs actuels est qu'ils tendent à générer très fréquemment des phrases génériques et peu informatives comme "Je ne sais pas" ou "Peut-être". Ce comportement découle de la diversité et de la rareté des phrases informatives, i.e. les

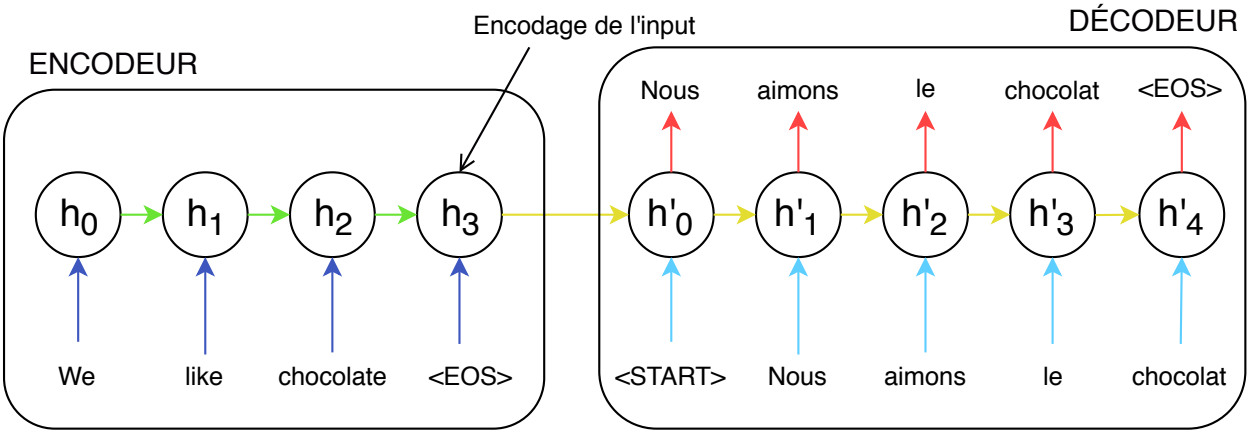


Figure 2.1 Architecture séquence-à-séquence. Dans cet exemple, le modèle doit traduire une phrase de l'anglais au français.

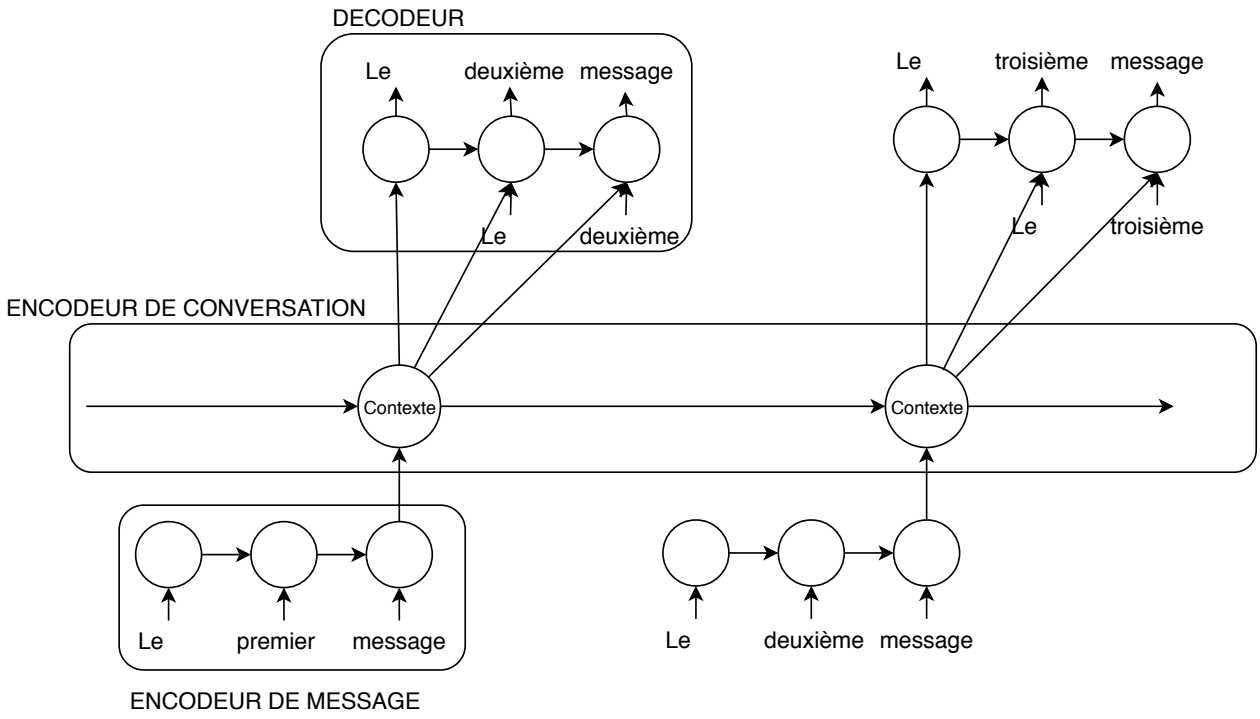


Figure 2.2 Architecture d'un Hierarchical Recurrent Encoder-Decoder (Sordoni et al., 2015).

phrases qui nous intéressent, par rapport aux phrases banales, dans les jeux de données de dialogue. Pour pallier ce problème, Li et al. (2015) reprend le concept de Maximisation de l'Information Mutuelle introduit par Bahl et al. (1986) qui mesure la dépendance mutuelle entre les inputs et les outputs. Suivant le schéma des auto-encodeurs variationnels (Kingma

and Welling, 2013), Serban et al. (2017a) introduit une variable latente à l'architecture d'encodeur-décodeur hiérarchique. Cette variable latente rend le décodeur stochastique et introduit de la diversité dans les réponses générées. Nous donnons plus de détails sur cette architecture en section 4.2.1.

Plusieurs travaux visent à incorporer des connaissances externes au modèle via des bases de connaissances, notamment pour le problème de *question answering*. Les *memory networks* sont une architecture spécialement adaptée à l'utilisation de connaissances externes dans le cadre d'une conversation lorsqu'il s'agit de répondre à des questions factuelles (Dodge et al., 2015; Ghazvininejad et al., 2017). L'approche des *memory networks* consiste à retrouver dans une mémoire, la base de connaissance, les énoncés permettant de répondre à la question posée. Dans le cas des recommandations, un *memory network* pourrait utiliser des méta-données sur les films pour produire des recommandations. Il est toutefois plus difficile d'imaginer des recommandations basées sur le comportement des utilisateurs comme le filtrage collaboratif car ici l'information est répartie sur les notes de tous les utilisateurs.

Méthodes d'extraction

Les méthodes d'extraction visent à choisir une réponse parmi un ensemble de candidats, le but étant de choisir la réponse le plus cohérente et appropriée. Lowe et al. (2015) encode le contexte actuel de la conversation, ainsi qu'une réponse candidate, et calcule la correspondance entre les deux. Puis, la réponse ayant la plus grande correspondance est choisie. Ces méthodes ne sont pas capables de générer de nouvelles phrases. Toutefois, elles ont l'avantage de toujours produire des phrases correctes et intelligibles. Par ailleurs, l'évaluation est plus simple que celle des modèles génératifs puisqu'il suffit d'examiner la précision.

2.1.3 Ensembles de modèles

De nombreux systèmes de dialogues à domaine ouvert (Krause et al., 2017; Serban et al., 2017b) combinent différents types de modules : certains modules pour répondre à des questions factuelles, d'autres pour bavarder du quotidien, ou encore pour parler de films, ou d'un autre domaine en particulier. Dans cette optique, Truong et al. (2017); Miller et al. (2017) ont développé des plate-formes facilitant l'intégration de tels systèmes de dialogues. Ces plate-formes permettent d'encapsuler les différents modules avec la même interface. Ces modules peuvent à leur tour être gérés par une composante de plus haut niveau.

2.2 Jeux de données de dialogue et de recommandation

Il n'existe pas de jeu de données de dialogues humains de recommandation de films rendu public à notre connaissance. Nous revoyons dans l'article en section 3.2 les jeux de données similaires, et en donnons plus de détails dans cette section. Pour une revue des corpus de dialogue en général, voir Serban et al. (2015).

Nos travaux se concentrent sur des dialogues de recommandation de films. Ce sont donc des dialogues à propos de films, à ne pas confondre avec des dialogues extraits de films (Banchs, 2012). Dodge et al. (2015) a introduit le Facebook Movie Dialog Data Set, qui comprend quatre jeux de données de dialogues à propos de films. Nous montrons dans le tableau 2.1 des exemples de dialogues tirés du Facebook Movie Dialog Data Set. Chacun de ces jeux de données a pour but de tester un aspect particulier d'un chat bot destiné à dialoguer à propos de films. Trois jeux de données sont synthétiques, générés grâce aux jeux de données *MovieLens* (Harper and Konstan, 2016) et *Open Movie Database*[1] : le *QA dataset*, le *Recommendation dataset* et le *QA + recommendation dataset*. Le *QA dataset* est constitué de dialogues d'un seul échange de question / réponse et teste la capacité d'un agent de dialogue à répondre à des questions factuelles. Les dialogues du *Recommendation dataset* ont un seul échange de recommandation de film. Ce jeu de données est similaire à celui que nous avons collecté, mais est malheureusement synthétique. De plus les dialogues ne sont constitués que d'un échange, et les réponses du recommandeur ne forment pas de prases complètes, ce sont juste des noms de films. Le *QA + recommendation dataset* propose des dialogues de trois échanges, i.e. trois réponses pour chaque participant, mélangeant recommandations et questions factuelles. Le quatrième jeu de données est constitué de conversations Reddit réelles tirées du subreddit *movies*[2]. Plusieurs participants peuvent prendre part à un sujet reddit. Pour simplifier, Dodge et al. (2015) le ramène à une conversation à deux participants : le parent, et les autres commentateurs du sujet. Les conversations Reddit présentent l'avantage d'être naturelles. Toutefois ce sont des discussions plus ouvertes, ne portant pas nécessairement sur des recommandations. Enfin, un jeu de données complémentaire combine de manière aléatoire les quatre tâches précédentes. Un agent de dialogue polyvalent sur le thème des films devrait être capable de produire des réponses pertinentes sur cette tâche. Il est connu qu'il est très difficile d'évaluer de manière automatique des modèles génératifs de langage (Liu et al., 2016). Dodge et al. (2015) propose plutôt d'évaluer des modèles capables d'ordonner différentes réponses possibles : étant donné une liste de candidats, dont un seul est la réponse humaine, la métrique *hit@k* vaut 1 si la vraie réponse est dans le top k, et 0 sinon. Si cette

1. http ://en.omdb.org
2. http ://reddit.com/r/movie

méthode ne permet pas d'évaluer les générations de modèles, elle permet d'évaluer le niveau de compréhension du dialogue qu'ont les différents modèles. Ils testent de nombreux modèles de référence comme des *bag-of-words*, des LSTM, des *memory networks* (Sukhbaatar et al., 2015), ainsi que d'autres modèles plus spécifiques aux recommandations ou au *question answering*.

Tableau 2.1 Exemples de dialogues tirés du Facebook Movie Dialog Dataset (Dodge et al., 2015). Le symbole <EOD> marque la fin d'un dialogue.

1 - Questions / réponses
Ruggero Raimondi appears in which movies ? Carmen <EOD>
What movies did Darren McGavin star in ? Billy Madison, The Night Stalker, Mrs. Pollifax-Spy <EOD>
2 - Recommandations
Schindler's List, The Fugitive, Apocalypse Now, Pulp Fiction, and The Godfather are films I really liked. Can you suggest a film ? The Hunt for Red October <EOD>
Some movies I like are Heat, Kids, Fight Club, Shaun of the Dead, The Avengers, Skyfall, and Jurassic Park. Can you suggest something else I might like ? Ocean's Eleven <EOD>
3 - QA et recommandations
I loved Billy Madison, My Neighbor Totoro, Blades of Glory, Bio-Dome, Clue, and Happy Gilmore. I'm looking for a Music movie. School of Rock
What else is that about ? Music, Musical, Jack Black, school, teacher, Richard Linklater, rock, guitar
I like rock and roll movies more. Do you know anything else ? Little Richard <EOD>
4 - Reddit
I think the Terminator movies really suck, I mean the first one was kinda ok, but after that they got really cheesy. Even the second one which people somehow think is great. And after that... forgeddabotit.
C'mon the second one was still pretty cool.. Arny was still so badass, as was Sararah Connor's character.. and the way they blended real action and effects was perhaps the last of its kind... <EOD>

Krause et al. (2017) ont collecté un jeu de données humain de *self-dialogues* pour la compétition *Amazon Alexa Prize* via la platforme Amazon Mechanical Turk (AMT). Étant donné un sujet, chaque participant doit imaginer une conversation entre deux personnes et jouer le rôle des deux interlocuteurs. Les trois sujets principaux sont la musique, les films et le sport. En dehors du sujet imposé, la conversation est libre, et n'a pas forcément pour sujet la recommandation. Même si la même personne joue le rôle des deux interlocuteurs, les dialogues sont de bonne qualité. Ce jeu de données constitue, à notre connaissance, un des seuls jeux de données publics de dialogue humain à propos de films. Ils utilisent ces données pour développer un agent de dialogue. Le but étant de produire un système éventuellement capable de dialoguer avec des humains, leur agent est composé de nombreux modules, chacun spécialisé dans un

type de réponse. Les trois principaux modules sont : un module qui utilise des règles définies manuellement, et qui est utilisé en cas de correspondance parfaite ; un module qui compare le contexte du dialogue avec les conversations du jeu de données pour reproduire des réponses vues dans le jeu de données ; et un réseau de neurones récurrent, générant une réponse si les deux autres modules n'en ont pas produit.

Dans sa thèse, Johansson (2004) collecte un jeu de données de 24 dialogues en tout point semblable au notre, si ce n'est par sa taille, malheureusement trop petite pour entraîner un modèle. Les conversations se font entre un participant qui joue le rôle de *recommandeur*, et un qui joue le rôle de *client* et cherche des recommandations de films. Le recommandeur a accès à une base de données de films. L'objectif pour le recommandeur est de suggérer au client cinq films que ce dernier a vus et aimés, et cinq qu'il n'a pas vus. Johansson (2004) utilise ces données pour analyser différents phénomènes qui apparaissent lors des dialogues de recommandation.

CHAPITRE 3 ARTICLE 1 : TOWARDS DEEP CONVERSATIONAL RECOMMENDATIONS

Authors: Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, Christopher Pal

Submitted and accepted by the conference: Neural Information Processing Systems 2018

Contribution: Implemented the models and ran the experiments with Hannes's insights. Co-implemented the data collection with Hannes and Samira. Important participation in the redaction and the bibliographic research.

Abstract

There has been growing interest in using neural networks and deep learning techniques to create dialogue systems. Conversational recommendation is an interesting setting for the scientific exploration of dialogue with natural language as the associated discourse involves goal-driven dialogue that often transforms naturally into more free-form chat. This paper provides two contributions. First, until now there has been no publicly available large-scale data set consisting of real-world dialogues centered around recommendations. To address this issue and to facilitate our exploration here, we have collected a data set consisting of over 10,000 conversations centered around the theme of providing movie recommendations. We intend to make this data available to the community for further research. Second, we use this dataset to explore multiple facets of conversational recommendations. In particular we explore new neural architectures, mechanisms and methods suitable for composing conversational recommendation systems. Our dataset allows us to systematically probe model sub-components addressing different parts of the overall problem domain ranging from: sentiment analysis and cold-start recommendation generation to detailed aspects of how natural language is used in this setting in the real world. We combine such sub-components into a full-blown dialogue system and examine its behavior.

3.1 Introduction

Deep learning-based approaches to creating dialogue systems provide extremely flexible machine learning-based solutions for the fundamental algorithms underlying dialogue systems. In this paper we explore fundamental algorithmic elements of conversational recommendation systems through examining a suite of neural architectures for sub-problems of conversational recommendation making. However, it is well known that deep learning techniques require considerable amounts of data to be effective. Addressing this need, we provide a new data set to the community to facilitate the study of discourse with natural language when making recommendations is an explicit goal of the exchange. Our setting of interest and our new dataset are centered around conversations about movies where one party in the conversation is seeking recommendations and the other party is providing recommendations. Our decision for focusing on this domain is motivated in part by the following.

A good discussion with a friend, librarian, movie rental store clerk or movie fan can be an enjoyable experience, leading to new ideas for movies that one might like to watch. We shall refer to this general setting as conversational movie recommendation. While dialogue systems are sometimes characterized as falling into the categories of goal-directed dialogue vs chit-chat, we observe that discussions about movies often combine various elements of chit-chat, goal-directed dialogue and even question answering in a natural way. As such the practical goal of creating conversational recommendation systems provides an excellent setting for the scientific exploration of discourse with natural language where generating recommendations is one of the explicit goals of the system.

This paper makes a number of contributions. First we provide the only real-world, two-party conversational corpus of this form (that we are aware of) to the community. We outline the data-collection procedure in Section 3.3. Second, we use this corpus to systematically propose and evaluate neural models for key sub-components of an overall conversational recommendation system. We focus our exploration on three key elements of such a system, consisting of: 1) Making recommendations; we examine sampling based methods for learning to make recommendations in the cold-start setting using an autoencoder (Sedhain et al., 2015). We present this model in Section 3.4.3 and evaluate it in Section 3.5. Prior work with such models has not examined the cold-start setting which must be addressed in our dialogue set-up. 2) Classifying opinions or the sentiment of a dialogue participant with respect to a particular movie. For this task throughout the dialogue whenever a new movie is discussed we instantiate an RNN-based sentiment-prediction model. This model is used to populate the autoencoder-based recommendation engine above. We present this model component and our analysis of its behavior and performance in Sections 3.4.2 and 3.5 respectively. 3) We

compose the components outlined above into a complete neural dialogue model for conversation and recommendation. For this aspect of the problem we examine a novel formulation of a hierarchical latent variable encoder-decoder (HRED) model (Serban et al., 2017a) with a switching mechanism inspired from (Gulcehre et al., 2016) that allows suggested movies to be integrated into the model for the dialogue acts of the recommender.

3.2 Related Work

While we are aware of no large scale public dataset of human-to-human dialogue on the subject of movie recommendations, we review some of the most relevant work of which we are aware below. We also review a selection of prior work on related methods in Section 4 just prior to introducing each component of our model.

Dodge et al. (2015) introduced four movie dialogue datasets comprising the Facebook Movie Dialog Data Set. There is a QA dataset, a recommendation dataset and a QA + recommendation dataset. All three are synthetic datasets built from the classic MovieLens ratings dataset (Harper and Konstan, 2016) and Open Movie Database [1]. Others have also explored procedures for generating synthetic dialogues from ratings data (Suglia et al., 2017). The fourth dataset is a Reddit dataset composed of around 1M dialogues from the movie subreddit[2]. The recommendation dataset is the closest to what we propose, however it is synthetically generated from natural language patterns, and the answers are always a single movie name. The Reddit data set is also similar to ours in the sense that it consists of natural conversations on the topic of movies. However, the exchanges are more free form and obtaining a good recommendation is not a goal of the discourse. Dodge et al. (2015) evaluate a ranked list of answers, given a list of candidates. They use the hit@k metric, that equals to 1 if the true answer is in the top-k, and 0 otherwise. They evaluate several baselines capable of ranking a list of candidate answers: memory networks (Sukhbaatar et al. (2015)), bag-of-word embedding models, LSTMs, and other models more specific to the tasks of question answering or recommendation. Krause et al. (2017) introduce a dataset of *self dialogues* collected for the Amazon Alexa Prize competition, using Amazon Mechanical Turk (AMT). The workers are asked to imagine a conversation between two individuals on a given topic and to play both roles. The topics are mostly about movies, music and sport. The conversations are not specifically about movie recommendations, but have the advantage of being quite natural, compared to the Facebook Movie Dialog Data Set. They use this data to develop a chat bot. The chat bot is made of several components, including: a rule-based component, a matching-score component that compares the context with similar conversations from the

1. [1]http://en.omdb.org, [2]http://reddit.com/r/movie

data to output a message from the data, and a (generative) recurrent neural network (RNN). They perform human evaluation of the matching-score component. Some older work from the PhD thesis of Johansson (2004) involved collecting a movie recommendation themed dialogue corpus with 24 dialogues, consisting of 2684 utterances and a mean of 112 utterances per dialogue. In contrast, our corpus has over 10k conversations and 160k utterances. See Serban et al. (2015) for a more recent survey of corpora for data-driven dialogue systems.

The recommender-systems literature has also proposed models for conversational systems. These approaches are goal-oriented and combine various different modules each designed (and trained) independently (Göker et al., 2011; Greco et al., 2017). Further, these approaches either rely on tracking the state of the dialogue using slot-value pairs (Widyantoro and Baizal, 2014; Wärnestål et al., 2007) or focus on different objectives such as minimizing the number of user queries to obtain good recommendations (Christakopoulou et al., 2016). In contrast, we propose a conditional generative model of (natural language) recommendation conversations and our contributed dataset allows one to both train sub-modules as well as explore *end-to-end* trainable models.

3.3 Data collection

Here we formalize the setup of a conversation involving recommendations for the purposes of data collection. To provide some additional structure to our data (and models) we define one person in the dialogue as the *recommendation seeker* and the other as the *recommender*. To obtain data in this form, we developed an interface and pairing mechanism mediated by Amazon Mechanical Turk (AMT). Our task setup is very similar to that used by Das et al. (2017) to collect dialogue data around an image guessing game, except that we focus on movie recommendations. We pair up AMT workers and give each of them a role. The movie seeker has to explain what kind of movie he/she likes, and asks for movie suggestions. The recommender tries to understand the seeker's movie tastes, and recommends movies. All exchanges of information and recommendations are made using natural language.

We add additional instructions to improve the data quality and guide the workers to dialogue the way we expect them to. Thus we ask to use formal language and that conversations contain roughly ten messages minimum. We also require that at least four different movies are mentioned in every conversation. Finally, we also ask to converse only about movies, and notably not to mention Mechanical Turk or the task itself. See Figure 3.4 in the supplementary material for a screen-shot of the interface.

In addition, we ask that every movie mention is tagged using the '@' symbol. When

workers type '@', the following characters are used to find matching movie names, and workers can choose a movie from that list. This allows us to detect exactly what movies are mentioned and when. Since most recent movies are not present in MovieLens, we gathered entities from DBpedia that were of type `<http://dbpedia.org/ontology/Film>` to obtain a list of movies. We also allow workers to add their own movies to the list if it is not present already. We obtained the release dates from the movie titles (e.g. `http://dbpedia.org/page/American_Beauty_(1999_film)`), or, if the movie title does not contain that information, from an additional SPARQL request. Note that the year or release date of a movie can be essential to differentiate movies with the same name, but released at different dates.

Table 3.1 Data set characteristics. For the movie dialogue forms, the numbers shown represent the seeker's answers.

# conversations	10021
# utterances	163820
# users	963
# movie mentions	51892
seeker mentioned	16338
recommender suggested	35554
not seen	16571
seen	31818
did not say	3503
disliked (4.9%)	2567
liked (81%)	42152
did not say (14%)	7173

Workers are (separately from the on-going discussion) asked three questions for each movie: (1) Whether the movie was mentioned by the seeker, or was a suggestion from the recommender ("suggested" label); (2) Whether the seeker has seen the movie ("seen" label): one of *Seen it*, *Haven't seen it*, or *Didn't say*; (3) Whether the seeker liked the movie or the suggestion ("liked" label): one of *Liked*, *Didn't like*, *Didn't say*. In the "liked" label, if the seeker has not seen the movie, then he must indicate whether he thinks it is a good suggestion or not. We will refer to these additional labels as *movie dialogue forms*. Both workers have to answer these forms even though it really concerns the seeker's movie tastes. Ideally, the two participants would give the same answer to every form, but it is possible that their answers do not coincide (because of carelessness, or dialogue ambiguity). The movie dialogue forms therefore allow us to evaluate sub-components of an overall neural dialogue system more systematically, for

Table 3.2 Conversation excerpts and model outputs. "SEEKER" refers to the seeker's utterances. "HUMAN" refers to the human recommender's utterances in the real dialogue. "HRED" and "OURS" refer to the generations of HRED and our model at that point in the dialogue. We provide additional conversation examples and model outputs in the supplementary material.

HUMAN:	hello, can i help you find a movie ?
HRED:	hi what kind of movies do you like
OURS:	hi, how can i help you ?
SEEKER:	hello ! i would like to find a suspenseful, but clean family friendly movie .
...	...
SEEKER:	2001 : a space odyssey might be a great option. anything else that you would suggest ?
HUMAN:	you can try planet of the apes the older one is quite suspenseful and family friendly .
HRED:	i haven't seen that one yet but i heard it was good. i haven't seen that one. have you seen the last house on the left ?
OURS:	star wars : the force awakens is also a good one return of the jedi all good movies
SEEKER:	those sound good ! i 'm going to look into those movies.
HUMAN:	i hope you enjoy, have a nice one
HRED:	have you seen foxcatcher ? it 's about a man who has a rich guy.
OURS:	i hope i was able to help you find a good movie to watch
SEEKER:	thank you for your help ! have a great night ! good bye

example one can train and evaluate a sentiment analysis model directly using these labels.

In each conversation, the number of movies mentioned varies, so we have different numbers of movie dialogue form answers for each conversation. The distribution of the different classes of the movie dialogue form is shown in Table 3.1. The liked/disliked/did not say label is highly imbalanced. This is standard for recommendation data (Marlin et al., 2007), since people are naturally more likely to talk about movies that they like, and the recommender's objective is to recommend movies that that the seeker is likely to like. Table 3.2 shows an example of conversation from the data set.

For the AMT HIT we collect data in English and chose to restrict the data collection to countries where English is the main language. The fact that we pair workers together slows down the data collection since we ask that at least two persons are online at the same time to do the task, so a good amount of workers is required to make the collection possible. Meanwhile, the task is quite demanding, and we have to select qualified workers. HIT reward and qualification requirement were decisive to get good conversation quality while still ensuring that people could get paired together. We launched preliminary HITs to find a compromise

and finally set the reward to $0.50 per person for each completed conversation (so each conversation costs us $1, plus taxes), and ask that workers meet the following requirements: (1) Approval percentage greater than 95, (2) Number of approved HITs greater than 1000, (3) Their location must be in United States, Canada, United Kingdom, Australia, or New Zealand.

3.4 Our Approach

We aim at developing an agent capable of chatting with a partner and asking questions about their movie tastes in order to make movie recommendations. One might therefore characterize our system as a recommendation "chat-bot". The complete architecture of our approach is illustrated in Figure 3.1. Starting from the bottom of Figure 3.1, there are four sub-components: (1) A general purpose sentence encoder based on the GenSen model (Subramanian et al., 2018) is used to produce a fixed length vector encoding for each sentence of the dialogue; (2) A novel variation of a hierarchical recurrent neural network based encoder-decoder (HRED) (Serban et al., 2016) architecture is used to model the dialogue acts generated by the recommender; (3) After each dialogue act our model detects if a movie entity has been discussed (with the @identifier convention) and we instantiate an RNN focused on classifying the seeker's sentiment or opinion regarding that entity. As such the sentiment analysis RNN is used as many times as there are movie entities discussed in the discourse. The sentiments regarding all the movies mentioned form the input to (4), an autoencoder based recommendation module. The autoencoder recommender's output is used by the HRED dialogue management module through a switching mechanism. We provide more details for each of these components below.

3.4.1 Our Hierarchical Recurrent Encoder

Our dialogue model is reminiscent of the hierarchical recurrent encoder-decoder (HRED) architecture proposed and developed in Sordoni et al. (2015) and Serban et al. (2016). The encoder follows the same hierarchical architecture, but we modify the decoder so that it can take explicit movie recommendations into account and we modify the encoder to take general purpose sentence (GenSen) representations arising from a bidirectional Gated Recurrent Unit (GRU) as input. Since our new dataset here consists of about 10k dialogues (which is relatively small for deep learning techniques), we use pre-trained GenSen representations obtained from the encoder outlined in Subramanian et al. (2018). These representations have led to higher performance across a variety of new tasks in lower data regimes (e.g. with only 10k examples). We use the embeddings and first layer of the GenSen sentence encoder which are pre-trained

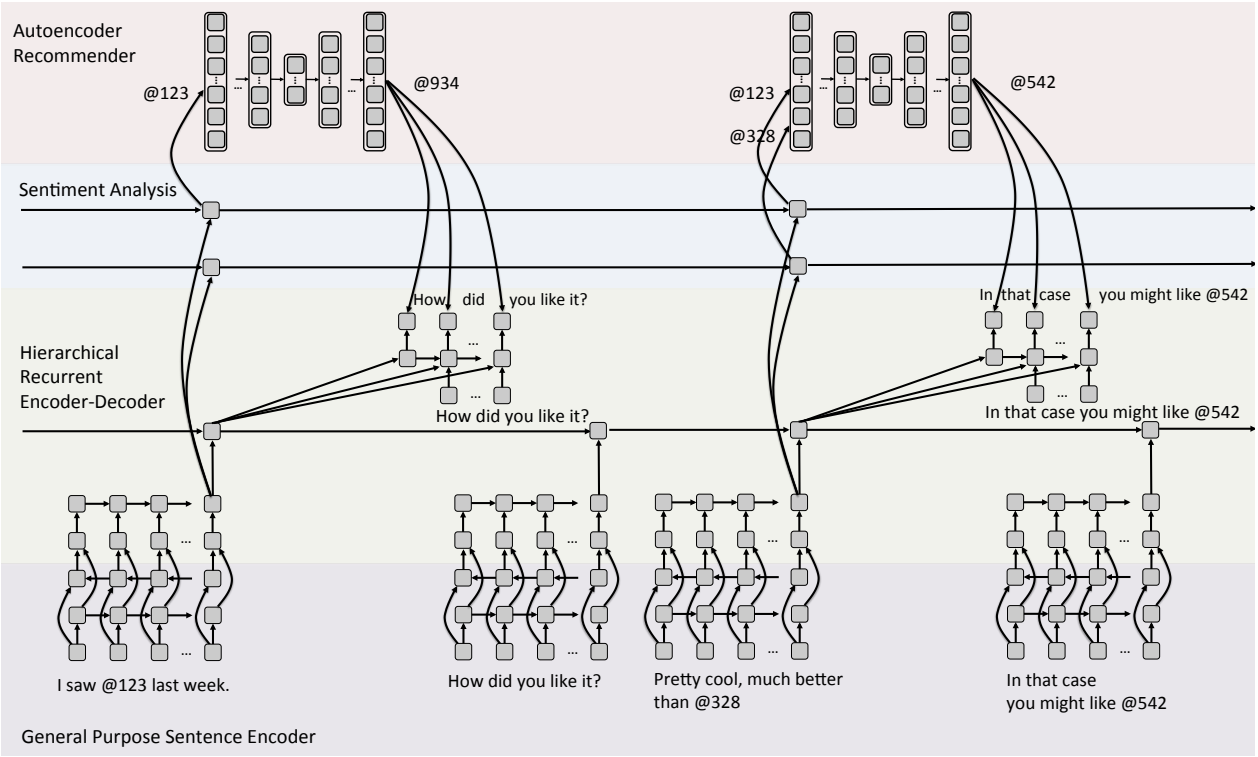


Figure 3.1 The complete architecture of our approach to modeling discourse for recommendation.

on multiple language tasks and we keep them frozen during training of our model. To deal with the issue of how to process movies discussed in the dialogue using the @movie for movie entities, @movie tokens in the input are replaced by the corresponding word tokens for the title of the movie.

More formally, we model each utterance U_m as a sequence of N_m words $U_m = (w_{m,1}, ..., w_{m,N_m})$ where the tokens $w_{m,n}$ are either words from a vocabulary V or movie names from a set of movies V'. We also use a scalar $s_1 \in \{-1, 1\}$ appended to each utterance to indicate the role (recommender or seeker) such that a dialogue of M utterances can be represented as $D = ((U_1, ..., U_M), s_1)$. We use a GRU to encode utterances and dialogues. Given an input sequence $(\mathbf{i}_1, ..., \mathbf{i}_T)$, the network computes reset gates r_t, input gates z_t, new gates n_t and forward hidden state $\overrightarrow{\mathbf{h}}_t$ as follows:

$$r_t = \sigma\left(\mathbf{W}_{ir} i_t + \mathbf{W}_{hr} \overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_r\right), \quad z_t = \sigma\left(\mathbf{W}_{iz} i_t + \mathbf{W}_{hz} \overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_z\right)$$
$$n_t = \tanh\left(\mathbf{W}_{in} i_t + \mathbf{b}_{in} + r_t(\mathbf{W}_{hn} \overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{hn})\right), \quad \overrightarrow{\mathbf{h}}_t = (1 - z_t) n_t + z_t \overrightarrow{\mathbf{h}}_{t-1}$$

Where the $\mathbf{W}_{**}$ and $\mathbf{b}_*$ are the learned parameters. In the case of a bi-directional GRU,

the backward hidden state $\overleftarrow{\mathbf{h}}_t$ is computed the same way, but takes the inputs in a reverse order. In a multi-layer GRU, the hidden states of the first layer $\left(\overrightarrow{\mathbf{h}}_1^{(1)}, ..., \overrightarrow{\mathbf{h}}_T^{(1)}\right)$ (or the concatenation of the forward and backward hidden states of the first layer $\begin{bmatrix}\overrightarrow{\mathbf{h}}_1^{(1)} \\ \overleftarrow{\mathbf{h}}_1^{(1)}\end{bmatrix}, ..., \begin{bmatrix}\overrightarrow{\mathbf{h}}_T^{(1)} \\ \overleftarrow{\mathbf{h}}_T^{(1)}\end{bmatrix}$ for a bi-directional GRU) are passed as inputs to the second layer, and so on. For the utterance encoder words are embedded in a 2048 dimensional space. Each utterance is then passed to the sentence encoder bi-directional GRU. The final hidden state of the last layer is used as utterance representation $\mathbf{u} = \begin{bmatrix}\overrightarrow{\mathbf{h}}_T^{(-1)} \\ \overleftarrow{\mathbf{h}}_T^{(-1)}\end{bmatrix}$. We obtain a sequence of utterance representations $\mathbf{u}_1, ..., \mathbf{u}_M$. To assist the conversation encoder we append a scalar $s_m \in \{-1, 1\}$ to each utterance representation $\mathbf{u}_m$, indicating if the sender is the seeker or the recommender. The sequence $\mathbf{u}'_1, ..., \mathbf{u}'_M$ is passed to the conversation encoder unidirectional GRU, which produces conversation representations at each step of the dialogue: $\mathbf{h}_1, ..., \mathbf{h}_M$.

3.4.2 Dynamically Instantiated RNNs for Movie Sentiment Analysis

To drive our autoencoder based recommendation module we build a model that takes as input both the dialogue and a movie name, and predicts for that movie the answers to the associated movie dialogue form. We remind the reader that both workers answer the movie dialogue form, but it only concerns the seeker's movie tastes. It often happens that the two workers do not agree on all the answers to the forms. It may either come from a real ambiguity in the dialogue, or from worker carelessness (data noise). So the model predicts different answers for the seeker and for the recommender. It outputs the three labels for each participant: the "suggested" label (binary output), the "seen" label (categorical output with three classes), the "liked" label (categorical output with three classes).

Let us denote $\mathcal{D} = \{(x_i, y_i), i = 1..N\}$ the training set, where $x_i = (D_i, m_i)$ is the pair of a dialogue D_i and a movie name m_i that is mentioned in D_i and

$$y_i = (\underbrace{y_i^{\text{sugg}}, y_i^{\text{seen}}, y_i^{\text{liked}}}_{\text{seeker's answers}}, \underbrace{y_i'^{\text{sugg}}, y_i'^{\text{seen}}, y_i'^{\text{liked}}}_{\text{recommender's answers}}), \tag{3.1}$$

are the labels in the movie dialogue form corresponding to movie m_i in dialogue D_i. So if 5 movies were mentioned in dialogue D, this dialogue appears 5 times in a training epoch.

The model is based on a hierarchical encoder (Section 3.4.1). For sentiment analysis, the model is modified at the utterance encoder level to take the movie m into account. After the first layer of the utterance encoder GRU (which is pre-trained), we add a dimension to the hidden states that indicate for each word if it is part of a movie mention. For example if we

condition on the movie *The Sixth Sense*, then the input ["<s>", "you", "would", "like", "the", "sixth", "sense", ".", "</s>"] produces the movie mention feature: [0, 0, 0, 0, 1, 1, 1, 0, 0]. The utterance and conversation encoding continue as described in Section 3.4.1 afterwards, producing dialogue representations $h_1, ..., h_M$ at each dialogue step.

The dialogue representation at the last utterance h_M is passed in a fully connected layer. The resulting vector has 14 dimensions. We apply a sigmoid to the first component to obtain the predicted probability that the seeker answered that the movie was suggested by the recommender o_i^{sugg}. We apply a softmax to the next three components to obtain the predicted probabilities for the seeker's answer in the not-seen/seen/did-not-say variable o_i^{seen}. We apply a softmax to the next three components to obtain the predicted probabilities for the seeker's answer in the disliked/liked/did-not-say variable o_i^{liked}. The last 7 components are treated the same way to obtain the probabilities of answers according to the recommender $o'^{\text{sugg}}, o'^{\text{seen}}, o'^{\text{liked}}$. We denote the parameters of the neural network by θ and $o_i = f_\theta(x_i) = \left(o_i^{\text{sugg}}, o_i^{\text{seen}}, o_i^{\text{liked}}, o_i'^{\text{sugg}}, o_i'^{\text{seen}}, o_i'^{\text{liked}}\right)$, the prediction of the model. We minimize the sum of the three corresponding cross-entropy losses.

Thereafter, only the seeker's liked prediction is used in the full model. However, we believe that training the model to predict all the labels may regularize it and improve its performance, as we show in section 3.5.

3.4.3 The Autoencoder Recommender

At the beginning of each conversation, the recommender has no prior information on the movie seeker (cold-start). During the conversation, the recommender gathers information about the movie seeker and (implicitely) builds a profile of the seeker's movie preferences. Sedhain et al. (2015) developed a user-based autoencoder for collaborative filtering (U-Autorec), a model capable of predicting ratings for users not seen in the training set. We use a similar model and pre-train it on the MovieLens dataset (Harper and Konstan, 2016).

We have m users, n movies and a partially observed user-movie rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. Each user $u \in \{1, ..., m\}$ can be represented by a partially observed vector $\mathbf{r}^{(u)} = (\mathbf{R}_{u,1}, ..., \mathbf{R}_{u,n})$. Sedhain et al. (2015) project $\mathbf{r}^{(u)}$ in a smaller space with a fully connected layer, then retrieve the full ratings vector $\hat{\mathbf{r}}^{(u)} = h(\mathbf{r}^{(u)}; \theta)$ with another fully connected layer. So during training they minimize the following loss:

$$L_{\mathbf{R}}(\theta) = \sum_{u=1}^{m} \|\mathbf{r}^{(u)} - h(\mathbf{r}^{(u)}; \theta)\|_{\mathcal{O}}^2 + \lambda \|\theta\|^2 \tag{3.2}$$

where $\| \cdot \|_{\mathcal{O}}$ is the L_2 norm when considering the contribution of observed ratings only and λ

controls the regularization strength.

To improve the performance of this model in the early stage of performing recommendations (i.e. in the so-called cold start setting) we train this model as a denoising auto-encoder (Vincent et al., 2008). We denote by N_u the number of observed ratings in the user vector $\mathbf{r}^{(u)}$. During training, we sample the number of inputs kept x uniformly at random in $\{1, ..., N_u - 1\}$. Then we draw x inputs uniformly without replacement among all the observed inputs in $r^{(u)}$, which gives us a noisy user vector $\tilde{\mathbf{r}}^{(u)}$. The term inside the sum of Equation 3.2 becomes $\|\mathbf{r}^{(u)} - h(\tilde{\mathbf{r}}^{(u)}; \theta)\|^2_{\mathcal{O}}$. The validation procedure is not changed: the complete input from the training set is used at validation or test time.

3.4.4 Our Decoder with a Movie Recommendation Switching Mechanism

Let us place ourselves at step m in dialogue D. The sentiment analysis RNNs presented above predict for each movie mentioned so far whether the seeker liked it or not using the previous utterances. These predictions are used to create an input $\mathbf{r}_{m-1} \in \mathbb{R}^{|V'|}$ for the recommendation system. The recommendation system uses this input to produce a full vector of ratings $\hat{\mathbf{r}}_{m-1} \in \mathbb{R}^{|V'|}$. The hierarchical encoder (Section 3.4.1) produces the current context $\mathbf{h}_{m-1}$ using previous utterances. The recommendation vector $\hat{\mathbf{r}}_{m-1}$ and the context $\mathbf{h}_{m-1}$ are used by the decoder to predict the next utterance by the recommender. The architecture of the model is shown in Figure 3.1.

For the decoder, a GRU decodes the context to predict the next utterance step by step. To select between the two types of tokens (words or movie names), we use a switch, as Gulcehre et al. (2016) did for the *pointer softmax*. The decoder GRU's hidden state is initialized with the context $\mathbf{h}_{m-1}$, and decodes the sentence as follows: $\mathbf{h}'_{m,0} = \mathbf{h}_{m-1}$, $\mathbf{h}'_{m,n} = \text{GRU}(\mathbf{h}'_{m,n-1}, w_{m,n})$, $\mathbf{v}_{m,n} = \text{softmax}\left(\mathbf{W}\mathbf{h}'_{m,n}\right)$, $\mathbf{v}_{m,n} \in \mathbb{R}^{|V|}$ is the predicted probability distribution for the next token $w_{m,n+1}$, knowing that this token is a word. The recommendation vector $\hat{\mathbf{r}}_{m-1}$ is used to obtain a predicted probability distribution vector $\mathbf{v}'_{m,n} \in \mathbb{R}^{|V'|}$ for the next token $w_{m,n+1}$, knowing that this token is a movie name: $\mathbf{v}'_{m,n} = \text{softmax}(\hat{\mathbf{r}}_{m-1}) = \mathbf{v}'_{m,0} \quad \forall n$. Where we note that we use the same movie distribution $\mathbf{v}'_{m,0}$ during the whole utterance decoding. Indeed, while the recommender's message is being decoded, it does not gather additional information about the seeker's movie preferences, so the movie distribution should not change. A switching network conditioned on the context $\mathbf{h}_{m-1}$ and the hidden state $\mathbf{h}'_{m,n}$ predicts the probability $d_{m,n}$ that the next token $w_{m,n+1}$ is a word and not a movie name.

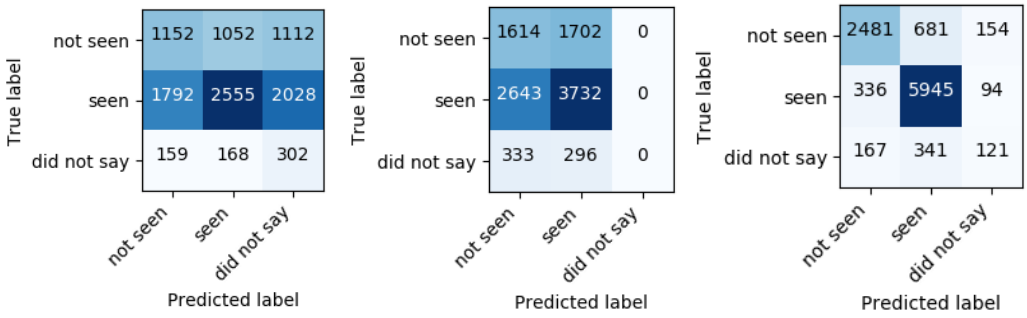
3.5 Experiments

We plan to continue the data collection with a separate pool of workers to form a test set. Thus, we split the 10021 conversations from the present data set into a training set and a validation set in a 80-20 proportion, and present the results on the validation set.

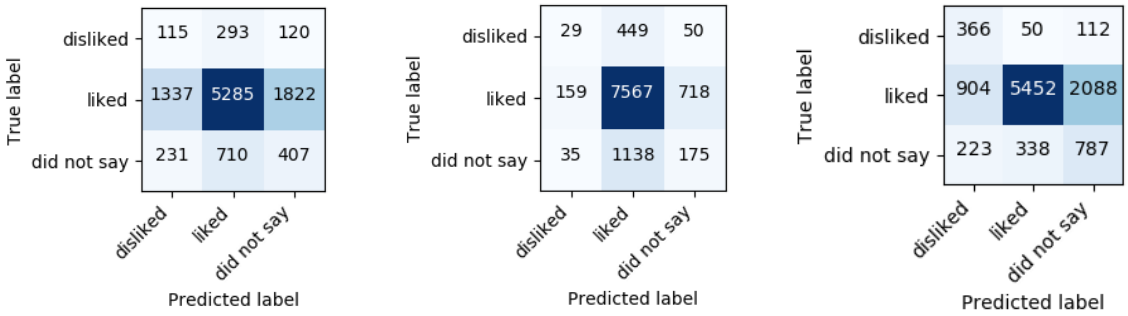
Movie sentiment analysis performance: We use the movie dialogue forms from our data to train and evaluate our proposed RNN based movie sentiment analysis formulation. The results obtained for the seeker's answers and the recommender's answers are highly similar, thus we present the results only for the seeker's answers. We focus on understanding if models are able to correctly infer the *seen* vs *not seen*, and *liked* vs *not liked* assessments from the forms. Because of the class imbalance (i.e. 81 % of movies were liked, vs 4.9 % which were disliked), we weight the loss to compensate for class imbalance. We compare with two simpler approaches. First, a baseline approach in which we pass the GenSen encodings of the sentences between the first and the last mention of a movie into a GRU layer. This is followed by a fully connected layer from the last hidden state. The prediction is made from the mean probability over all the sentences. Secondly, instead of using a single hierarchical encoder that is jointly trained to predict the three labels (suggested, seen and liked), we train the same model with only one of the three labels (either seen or liked) and demonstrate that the joint training regularizes the model. Figures 3.2a and 3.2b show the confusion matrices for the *seen* and *liked* prediction tasks for, from left to right, the baseline model, our model trained on single objectives, and our method outlined in Section 3.4.2 and illustrated in the blue region of Figure 3.1. We also provide Cohen's kappa coefficient (Cohen, 1960) for each model and each prediction task. Cohen's kappa measures the agreement between the true label and the predictions, and is thought to be more robust than a simple accuracy, notably in a case of imbalanced classes, where the probability of agreeing by chance if very high. For each prediction task, our jointly trained model has a higher kappa coefficient than the two other baselines. The full confusion matrix for the Cartesian product of predictions is shown in Figure 3.2c. All results are on the validation set.

Table 3.3 RMSE for movie recommendations. RMSE is shown for ratings on a 0-1 scale. For the MovieLens experiment, we show the RMSE on a 0.5-5 scale in parenthesis.

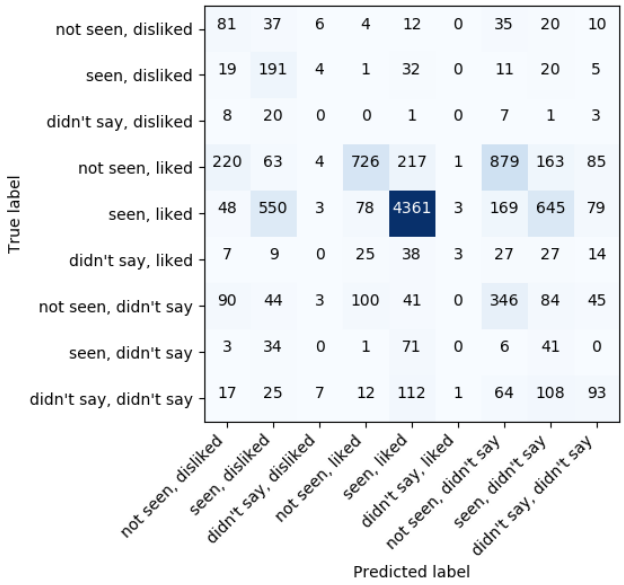
		Experiments on our data (validation RMSE)	
Training procedure	Experiments on MovieLens	No pre-training	Pre-trained on MovieLens
Standard Baseline	0.181 ± 0.001 (0.813)	0.125	0.075
Denoising Autorec	$\mathbf{0.171} \pm 0.0006$ (0.769)	0.127	**0.072**



(a) Confusion matrices for the *seen* label. (left column) Baseline GRU experiment $\kappa = 0.069$ (middle) Our method with separate objectives $\kappa = 0.062$ (right) Our method, jointly trained $\kappa = 0.65$



(b) Confusion matrices for the *liked* label. (left column) Baseline GRU experiment $\kappa = 0.056$ (middle) Our method with separate objectives $\kappa = 0.054$ (right) Our method, jointly trained $\kappa = 0.27$



(c) Confusion matrix for the Cartesian product predictions of *seen* and *liked* labels using our method.

Figure 3.2 Confusion matrices for movie sentiment analysis on the validation set. We also provide Cohen's kappa coefficient for each matrix.

Movie recommendation quality: We use the "latest full" MovieLens data set, that contains 26 million ratings across 46,000 movies, given by 270,000 users. It contains 2.6 times more ratings, but also across 4.6 times more movies than MovieLens-10 M, the data set used in Sedhain et al. (2015). In a first experiment we evaluate the model on the MovieLens data set. As did Sedhain et al. (2015), we sampled the training, validation and test set in a 80-10-10 proportion, and repeated this splitting procedure five times, reporting the average Root-Mean-Square Error (RMSE). We also examine how the model performs on the ratings from our data, with and without pre-training on MovieLens. We chose to consider only the ratings given by the movie seeker, and to ignore the responses where the user answered "did not say either way". We end up with a set of binary ratings for each conversation. To place ourselves in the setting of a recommender that meets a new movie seeker, we consider each conversation as a separate user. Since the ratings in the MovieLens are not binary, but range between 0.5 and 5, we binarize it for the pre-training, chosing a threshold that gives a similar distribution of 0 and 1 as in our data. Knowing that our data has 94.3% of "liked" ratings, we chose a rating threshold of 2: ratings higher or equal are considered as "liked", ratings lower are considered as "disliked". The binarized MovieLens data set now has 93.7% of "liked" ratings. In each experiment, for the two training procedures (standard and denoising), we perform an hyper-parameter search on the validation set. Table 3.3 shows the RMSE obtained on the test set. In the experiment on the MovieLens data set, the denoising training procedure brings a slight improvement on the standard training procedure. After pre-training on MovieLens, the performances of the models on our data is significantly improved.

Overall dialogue quality assessment: The recommendations provided by the whole model are based on the recommender module, evaluated in the previous paragraph. Liu et al. (2016) has shown that the most reliable way to evaluate generative dialogue models is human evaluation. Thus, we perform a user study to assess the overall quality of the dialogue text generated with our baseline HRED model versus our new model. Ten participants were each presented with ten complete real dialogues from our validation set, performing 56 ranking tasks. At the point where the human recommender provided their response in the real dialogue we show: the text generated by our HRED baseline, our model and the true response in a random order. The participant is asked to give the dialogue responses a rank from 1–3, with 1 being the best and 3 the worst. We allow ties so that multiple responses could be given the same rank (ex. rankings of the form 1, 2, 2 were possible if the one response was clearly the best, but the other two were of equivalent quality). We show the percentage of times that each model was given each ranking. The true response was ranked #1 349 times, our model 267 times, our HRED baseline was ranked #1 223 times.

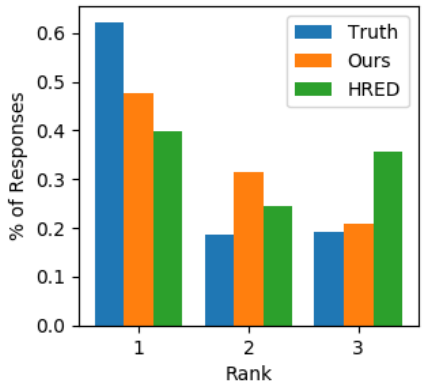
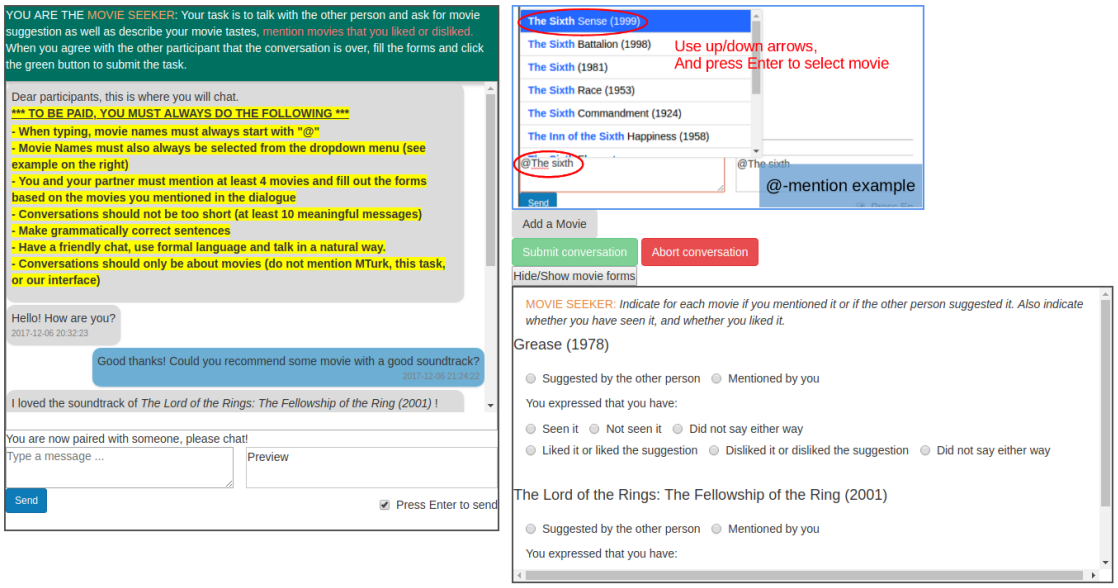


Figure 3.3 Results of human assessment of dialogue quality.

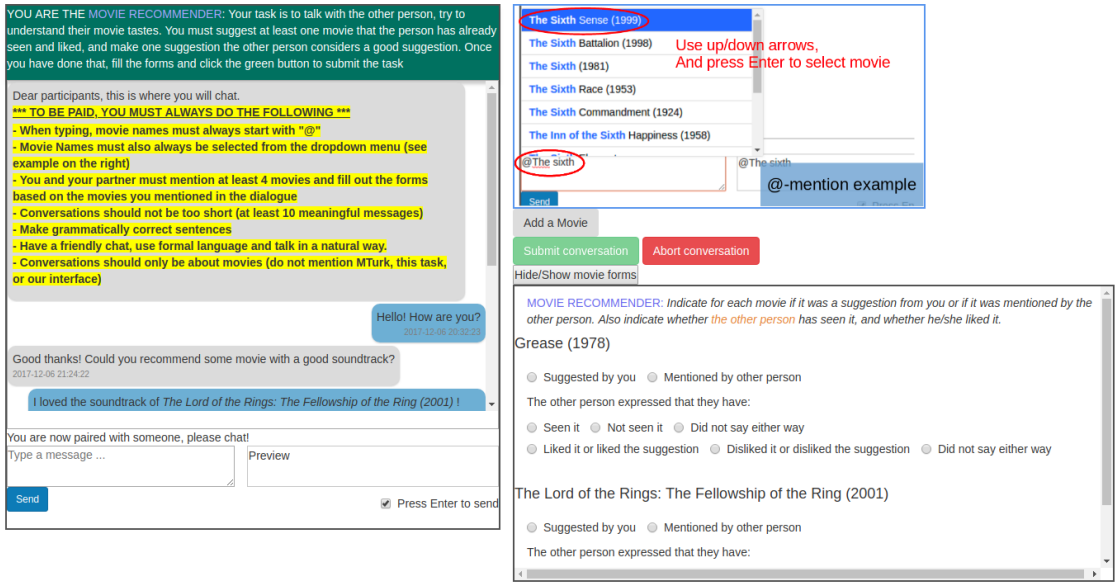
3.6 Discussion and Conclusions

We have presented a new, high utility dataset of real-world, human generated conversations around the theme of providing movie recommendations and we have used this dataset to explore a novel modular formulation of a fully neural architecture for conversational movie recommendations. The dataset has been collected in such a way that subtasks such as sentiment analysis and movie recommendation can be explored and evaluated separately or within the context of a complete dialogue system. We have presented a novel overall architecture for this problem domain which leverages general purpose sentence representations and hierarchical encoder-decoder architectures, extending them with dynamically instantiated RNN models that drive an autoencoder based recommendation engine. We find tremendous benefit from this modularization in that it allows one to pre-train the recommendation engine on other larger data sources specialized for the recommendation task alone. Further, our proposed switching mechanism allows one to integrate recommendations within a recurrent decoder, mixing high quality suggestions into the overall dialogue framework.

Appendix



(a) Seeker interface



(b) Recommender interface

Figure 3.4 Data collection interface.

We show in tables 3.4, 3.5, 3.6, 3.7, 3.8, some randomly picked conversations from the validation set, and the responses generated by the HRED and by our model. At each utterance, we condition on the ground truth previous turns (labeled “SEEKER” and “HUMAN” in the

tables), and use a beam search of size 10 to generate the response.

Table 3.4 Sample conversation from validation set.

SEEKER:	hi ! i 'm looking for an action filled movie similar to jurassic park
HUMAN:	ok , have you seen jurassic world ?
HRED:	hi ! i love that one ! have you seen the conjuring or the conjuring 2 ?
OURS:	hi what kind of movies do you like ?
SEEKER:	yes i loved that movie as well . are there any good movies without dinosaurs ?
HUMAN:	have you seen king kong ?
HRED:	no , i have n't seen it yet . have you seen the avengers ? it 's about a couple of friends who are their own .
OURS:	i have n't seen that one . i will have to check it out . have you seen jurassic park
SEEKER:	yes ! that was another great giant monster movie :)
HUMAN:	what about jaws & amp ; jaws 2 ? oldies but goodies scary action ? suspense gives me the rumblies
HRED:	i have n't seen that one yet but i heard it was good . i have n't seen it yet . i have n't seen it yet . i have n't seen it yet .
OURS:	i have n't seen that one yet . i will have to check it out . what about walking with dinosaurs ?
SEEKER:	i absolutely loved the first one . one of my favorite movies ever :) honestly i ca n't say i remember much about the second one . jaws 3-d was quite funny and jaws : the revenge was just weird . i do love suspense too ...
HUMAN:	i like suspense but sometimes i can barely handle it ! it gives me anxiety ! lol
HRED:	no , i have n't seen it yet . i will have to check it out . thanks for the chat .
OURS:	i have n't seen that one . i will have to check it out . i hope i helped .
SEEKER:	that 's understandable . as long as it 's not too gory i do n't generally have a problem with those things .
HUMAN:	well , it was great chatting with you ! have a great one !
HRED:	i 'm glad i was able to help you today
OURS:	have you seen immortals ? it 's a war movie but it 's pretty good
SEEKER:	you too ! thanks !

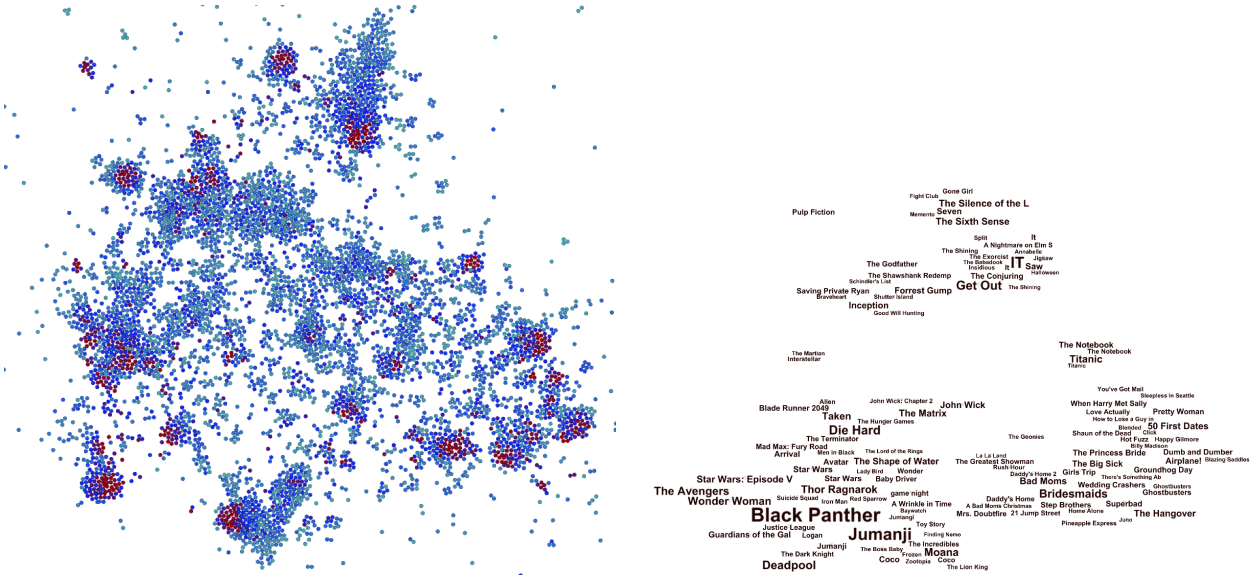


Figure 3.5 2D embedding of movies in our conversation database. The edge weight in the similarity matrix is proportional to the number of co-occurrences in the same dialogue. Left: all movies, colored by number of occurrences from light blue (low) to red (high). Right: names of movies with highest number of occurrences. Embedding via Jacomy et al. (2014).

Table 3.5 Sample conversation from validation set

HUMAN:	hello .
HRED:	hi what kind of movies do you like
OURS:	hi , how can i help you ?
SEEKER:	hi
HUMAN:	what type of movie can i suggest to you ?
HRED:	what kind of movies do you like ? i 'm good
OURS:	what kind of movies do you like ? what kind of movies do you like ?
SEEKER:	i am looking for a slap stick comedy from the 80 's or 90 's
HUMAN:	not familiar with this , please give me a minute .
HRED:	have you seen the conjuring or the conjuring 2 ? it was really good and very funny
OURS:	what kind of movies do you like ? i 'm good , thanks for asking ! have a great day !
SEEKER:	okay movies like airplane ! and airplane ii : the sequel
HUMAN:	oh , okay . i saw those and thought they were great . did you see caddyshack or dumb and dumber ?
HRED:	i have n't seen that one yet but i heard it was good . have you seen the last house on the left ? it 's pretty good .
OURS:	have you seen coming to america ? or the naked gun hot shots !
SEEKER:	yes , those were pretty good
HUMAN:	how about spaceballs , i thought this was pretty funny .
HRED:	have you seen foxcatcher ? it 's a comedy about a man on a true story
OURS:	have you seen dumb and dumber to ? it 's a classic !
SEEKER:	haha that one was great
HUMAN:	okay , let ' me see if i can offer one more .
HRED:	have you seen foxcatcher ? it 's a comedy about a man that turns out on a true story .
OURS:	i have n't seen that one yet . i 'll have to look it up .
SEEKER:	awesome
HUMAN:	robin hood : men in <unk> ? this one had me rolling .
HRED:	have you seen foxcatcher ? it 's about a man who has a rich guy who is a <unk> rich guy
OURS:	have you seen monty python 's life of brian ? that 's a good one !
SEEKER:	haha i forgot about that one ! i 'll be watching that one tonight
HUMAN:	great .
HRED:	it stars nanjiani , zoe kazan , holly hunter , ray romano , adeel akhtar , and anupam <unk> no , i have n't seen it yet .
OURS:	have you seen caddyshack ii ? it 's a good one .
SEEKER:	thanks for all of your help i hope you have a great day
HUMAN:	welcome , glad to help . have a nice day . good bye
HRED:	you 're welcome ! have a great day !
OURS:	you 're welcome ! have a great day !

Table 3.6 Sample conversation from validation set

SEEKER:	hi !
HUMAN:	hi ! how is your morning going ?
HRED:	hi , what kind of movies do you like
OURS:	hello ! what kind of movie are you looking for ?
SEEKER:	great , thanks ! i was thinking of starting the day off with a great comedy , any suggestions ?
HUMAN:	i love the hangover it always makes me laugh . and i bet lots of people are hungover this morning ! airplane ! is another classic of mine .
HRED:	that 's a good one ! have you seen the lego movie ? it 's a great movie !
OURS:	i have n't seen that one ! what kind of movies do you like ? i can help you with that .
SEEKER:	i 've seen the hangover it 's really good ! i thought about watching knocked up have you seen it ?
HUMAN:	i enjoyed that movie . i enjoyed spy it 's a well done parody .
HRED:	it 's about a man who gets revenge on a cyborg and a hockey team . it 's really good . it 's really good . i also like the girl on the train
OURS:	i have n't seen that one . i will have to check it out . i hope i was able to help you find a good one ! have a great day !
SEEKER:	thanks ! i 'll check it out !
HUMAN:	sure ! have a great day !
HRED:	you 're welcome ! have a great day !
OURS:	have you seen revenge of the nerds ? it 's really good . it 's a classic .
SEEKER:	you too .

Table 3.7 Sample conversation from validation set

HUMAN:	good day !
HRED:	hi what kind of movies do you like
OURS:	hi , how can i help you ?
SEEKER:	hey i 'm really into romance movies any good suggestions ? thanks !
HUMAN:	hmm . let me think for a sec . have you seen the big sick ? it 's romantic and funny
HRED:	what kind of movies do you like to watch ? have you seen the conjuring 2 ?
OURS:	what kind of movies do you like ? i 'm good , and you ?
SEEKER:	i like movies like beauty and the beast and a walk to remember oh and lol i have never seen the the big sick or never been kissed and romeo and juliet
HUMAN:	i highly recommend ! great story ! have you seen the notebook ? that 's a big one <unk> romance fans
HRED:	i have n't seen that one yet but i 've heard good things about it !
OURS:	i have n't seen that one yet but i have heard good things about the longest ride and the fault in our stars
SEEKER:	i love the the notebook a really great film .
HUMAN:	and if you liked romeo and juliet then i have to reccommend titanic for leo . also a recent sweet romantic comedy was the duff
HRED:	i have n't seen that one , but i 've heard good things about it !
OURS:	i have n't seen that one yet but i 've heard good things about it have you seen message in a bottle ?
SEEKER:	ahh the great titanic what a classic thanks for recommending that one . i will definitely have to check out the the duff thanks so much for your help !
HUMAN:	no problem ! i have one more to suggest . phantom thread ! it 's new and i did n't like it as much but a lot of others do ! it <unk> romance in a unique way no problem at all ! i hope you enjoy the films you decide to watch and enjoy the rest of your day !
HRED:	you 're welcome ! have a great day !
OURS:	you 're welcome ! have a great night !
SEEKER:	thanks i will defiantly check out the phantom thread i really appreciate it and you have a great day !

Table 3.8 Sample conversation from validation set

SEEKER:	hello
HUMAN:	hello
HRED:	hi , how can i help you ?
OURS:	hello , what kind of movie are you looking for ?
SEEKER:	how are you this evening ?
HUMAN:	what type of movies do you like ? good , thanks
HRED:	what kind of movies do you like ? i 'm good
OURS:	can i help you find a movie ?
SEEKER:	i am looking for a musical . what do you suggest ?
HUMAN:	have you seen enchanted ? maybe you 'd like la la land .
HRED:	have you seen get out yet ? it 's pretty good
OURS:	i have n't seen that one yet but have heard good things about it have you seen prometheus ?
SEEKER:	no i have n't seen that one . i have been meaning to watch it though . oh that 's a good suggestion . what about something older ?
HUMAN:	the rocky horror picture show is a great one .
HRED:	i have n't seen that one yet but heard good things about ender 's game i have n't seen it yet but i have n't seen it yet but i 've heard of it .
OURS:	i 'm trying to think of something along the same way . sweeney todd : the demon barber of fleet street is another good one
SEEKER:	i 've seen that before when i was younger . i did n't know whether to laugh or be scared . lol
HUMAN:	fiddler on the roof maybe ? sorry , i have n't seen many musicals
HRED:	i have n't seen that one . i will have to check it out . thanks for chatting !
OURS:	i have n't seen grease but i have heard good things about it .
SEEKER:	yes ! that 's great ! thank you for the suggestions ! i try to watch them soon , especially enchanted ! hope you have a good night !
HUMAN:	thanks you too
HRED:	you 're welcome ! have a great night !
OURS:	no problem ! have a good day !

CHAPITRE 4 DISCUSSION GÉNÉRALE ET AUTRES CONTRIBUTIONS

4.1 Discussion de l'article

Dans cette partie, nous revenons sur quelques détails de l'article présenté en section 3 et expliquons certains choix (qui n'ont pas été expliqués par manque de place). Nous décrivons d'abord la récolte des données plus en détails (section 4.1.1), puis nous donnons des statistiques pertinentes du jeu de données (section 4.1.2). En section 4.1.3, nous revenons sur le procédé de génération de phrases. Enfin, nous évaluons la possibilité de continuer la récolte des données (section 4.1.4).

4.1.1 Retour sur la récolte des données

Cette partie s'intéresse à la récolte des données décrite en section 3.3.

Plateforme Amazon Mechanical Turk AMT est une plate-forme développée par Amazon permettant de mettre en lien des *demandeurs* (*requesters* en anglais) et des *participants*, ou *travailleurs* (*workers*). Les demandeurs sont des personnes ou institutions ayant besoin de récolter des données, ou bien de les labéliser. Il s'agit en général d'une tâche répétitive à effectuer en quantité massive, mais pas réalisable par une machine dans l'état actuel de la technologie. Le demandeur poste la tâche à réaliser sur AMT, puis les travailleurs peuvent se rendre sur AMT pour la réaliser, moyennant une rémunération. Le travail sur AMT se quantifie en Human Intelligence Task (HIT). Un HIT représente l'unité de travail qui sera rémunérée. Dans notre cas, compléter un HIT revient à terminer une conversation avec un partenaire et à répondre à tous les *movie forms*. Une fois le HIT envoyé, le demandeur a pour devoir d'*approuver* le travail produit par chaque participant, ou bien de le *rejeter* si c'est justifié. AMT réalise cette interface entre les demandeurs et les travailleurs. De plus en plus de projets, notamment en apprentissage machine, utilisent cette plate-forme pour créer ou labéliser des jeux de données. En tant que demandeurs, nous avons utilisé AMT pour récolter notre jeu de données.

Obtention d'une liste de films Puisque Movielens ne comprend pas les films sortis après août 2017, nous avons préféré extraire une list de titres de fims de DBpedia [1] comme décrit en section 3.3. Certains noms de films ne sont pas exactement les mêmes dans Movielens

1. `https://wiki.dbpedia.org/`

et dans DBpedia. Par exemple, “The Prince and Me” dans DBpedia devient “Prince & Me, The” dans Movielens. Un script qui utilise des règles définies manuellement nous permet de faire correspondre 4697 films sur les 6573 films mentionnés au moins une fois dans nos données. Parmi les 1876 films restants , il nous est impossible de savoir combien auraient dû avoir une correspondance. Toutefois, nous avons constaté en section 3.5 que le taux de films appareillés était suffisant pour effectuer un pré-entraînement utile. Dans le cas où le titre du film ne contient pas sa date de sortie, cette dernière peut souvent être obtenue en ajoutant à la requête SPARQL :

```
OPTIONAL {?title    dct:subject     ?subject.
          ?subject  skos:broader    dbc:Films_by_year.}
```

À propos de la gestion des participants Un facteur crucial de cette récolte de données fut le nombre de participants. Comme discuté en section 3.3, l’ajustement du prix des HIT et des qualifications requises influent grandement sur la qualité des conversations collectées et le nombre de participants. Nous avons souvent observé que certains participants qui apprécient le HIT en complètent un très grand nombre en peu de temps. Dans le but d’obtenir un jeu de données aussi diversifié que possible, nous voulons éviter que quelques travailleurs consomment la totalité des HIT publiés : nous avons donc limité à 100 le nombre total de conversations pour chaque travailleur. Cette limite s’applique seulement une fois que notre équipe a approuvé 100 conversations pour un participant donné. Ainsi, avec le délai nécessaire pour approuver les conversations, certains participants ont facilement pu dépasser les 100 conversations, allant jusqu’à 232 au maximum.

Afin d’attirer le plus de participants possibles, il est essentiel qu’ils apprécient le travail qu’on leur propose, d’autant que le forum Turkopticon [2] permet aux travailleurs de noter et commenter les différentes tâches. Cela passe entre autres par une rémunération juste et équitable selon le standard d’AMT. Notre tâche est relativement complexe comparée aux tâches habituellement proposées sur AMT (la tâche fait intervenir deux participants, il faut réfléchir aux recommandations de films, discuter dans un anglais correct, remplir un formulaire avant d’envoyer la conversation). Il arrive donc que le travail effectué ne réponde pas aux instructions fournies. Sur tous les HITs envoyés, entre le 1er décembre 2017 et le 19 avril 2018, nous en avons approuvés 96.7%. Il est très important que chaque rejet soit justifié et expliqué au participant. La vérification de toutes les conversations récoltées constitue un travail phénoménal, qui a été fait avec minutie par des membres de l’équipe des éditeurs de Maluuba,

2. https://turkopticon.ucsd.edu/

coordonnée par Adam Ferguson. Nous remercions chaleureusement toute cette équipe pour leur contribution au projet. Dans une moindre mesure, le contact avec les participants par courrier électronique est également important, dans le cas où un rejet est incompris ou injustifié.

Jeu de test L'article présente en section 3.5 tous les résultats d'expériences sur le jeu de validation. En effet, le jeu de test n'était pas encore récolté à l'époque. Depuis la soumission de l'article, nous avons continué à lancer des HITs pour récolter un jeu de test. Ainsi, entre le 5 mai 2018 et le 15 juin 2018, nous avons récolté 1342 conversations de test. Pour s'assurer que ces données soient indépendantes des données d'entraînement, nous récoltons ces données avec seulement des nouveaux participants. À l'avenir, ce jeu de test pourra servir de base pour comparer les résultats de différents modèles.

4.1.2 Statistiques du jeu de données

Nous montrons ici quelques statistiques intéressantes du jeu de données d'entraînement pour en comprendre un peu mieux la composition. Nous ne considérons que les conversations qui ont été approuvées et qui font donc partie du jeu de données final. Ainsi, ne sont pas comprises les conversations qui ont été interrompues avant d'être envoyées, ou les conversations ne répondant pas entièrement à nos critères.

Sur les 10021 conversations d'entraînement, 163820 messages ont été échangés entre 963 participants distincts. Les figures 4.1 et 4.2 montrent les histogrammes des longueurs des conversations et des messages. On constate que ces deux distributions ont une queue longue, avec de très fortes fréquences pour des longueurs relativement faibles (autour de 15 messages par conversations, et 15 mots par message), et une fréquence qui décroît exponentiellement vers zéro. La plus longue conversation comporte 83 messages, et le plus long message 183 mots. 6573 films distincts ont été mentionnés, pour un total de 51892 mentions de films. À titre de comparaison, la liste de films récoltée à partir de DBpedia contient 128438 films, et Movielens comprend 45843 films. Le nombre de films distincts mentionnés au cours de la récolte augmente de moins en moins, comme le montre la figure 4.3. C'est un phénomène attendu, puisque plus le nombre de films déjà mentionnés est important, plus il est difficile de mentionner de nouveaux films. Toutefois le nombre de films distincts mentionnés augmente toujours de manière soutenue à la fin de la collecte de données. On peut donc supposer qu'un large spectre de films n'a toujours pas été exploré. La figure 4.4 donne l'histogramme du nombre de mentions par film. On observe une distribution très similaire à celle qu'on obtiendrait pour la fréquence d'utilisation des mots dans un corpus donné, une loi de Zipf.

La figure 4.5 est un nuage de points où chaque point correspond à un film mentionné au moins

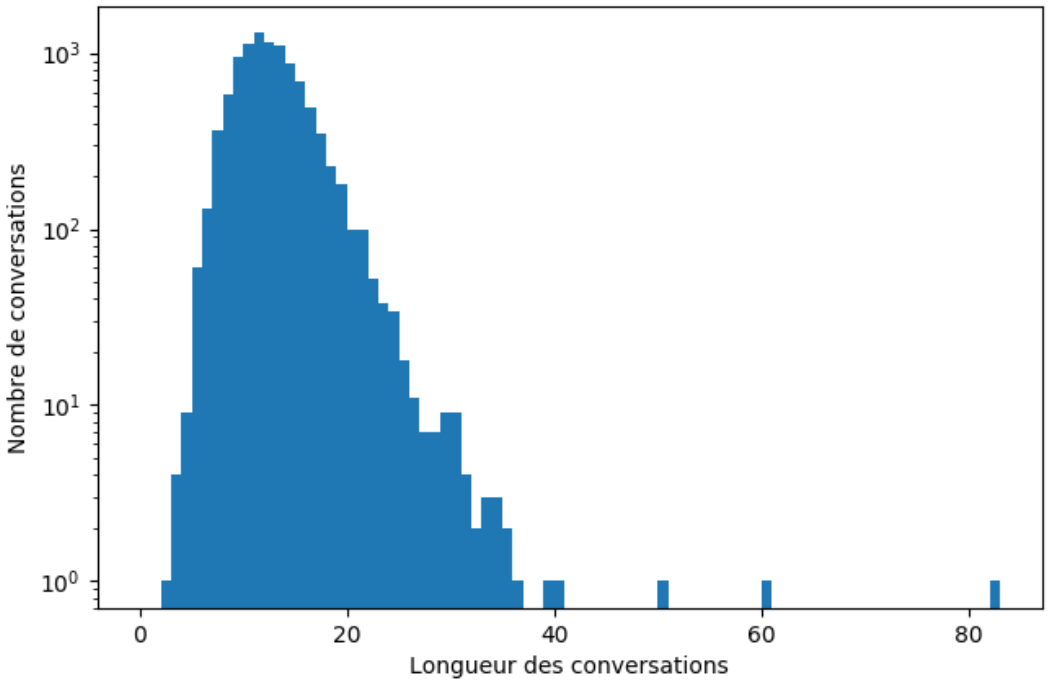


Figure 4.1 Histogramme de la longueur des conversations, en nombre de messages. Axe vertical en échelle logarithmique.

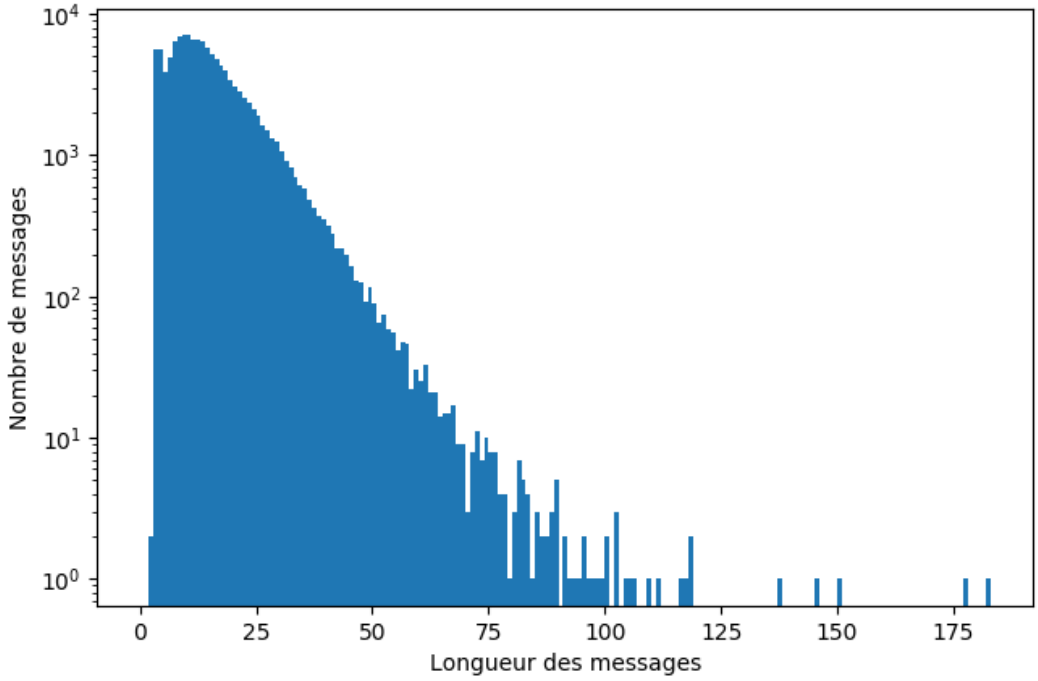


Figure 4.2 Histogramme de la longueur des messages échangés, en nombre de mots. Axe vertical en échelle logarithmique.

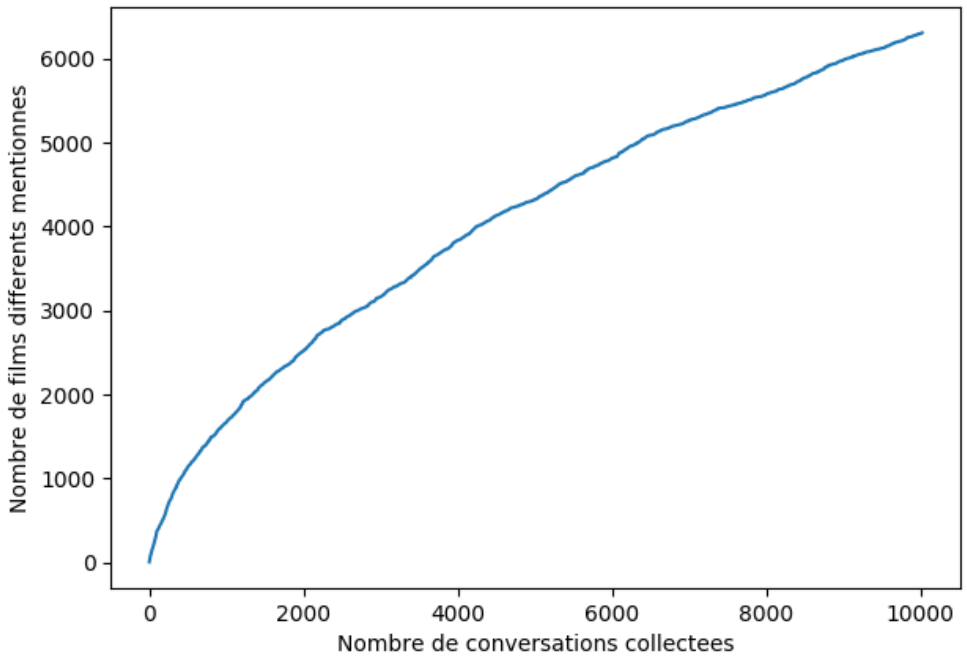


Figure 4.3 Nombre de films distincts mentionnés en fonction du nombre de conversations collectées.

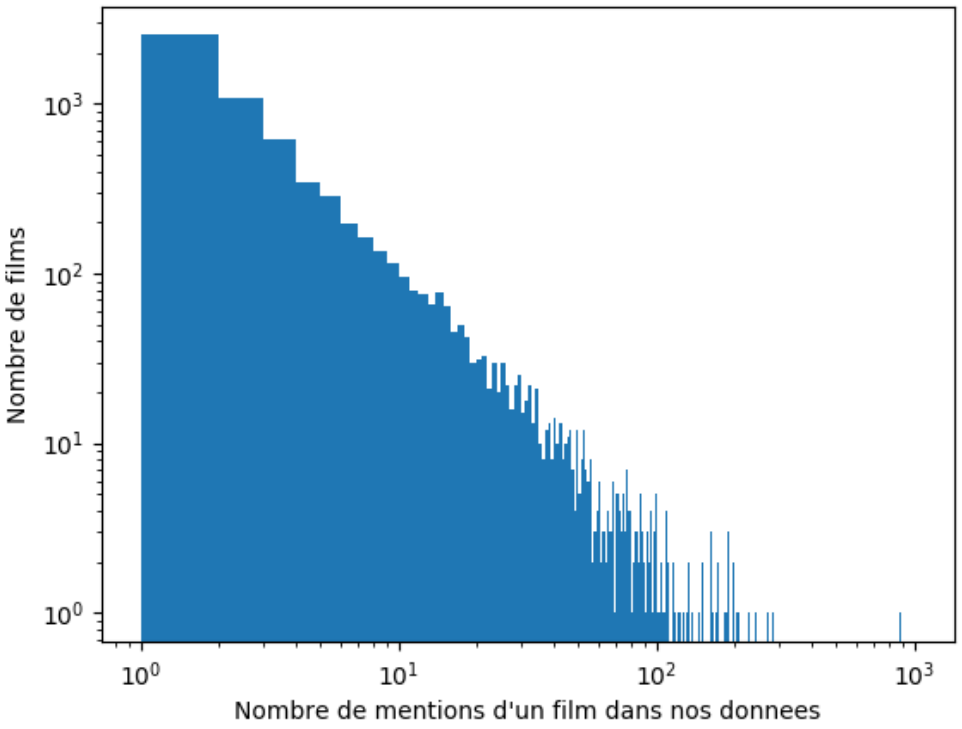


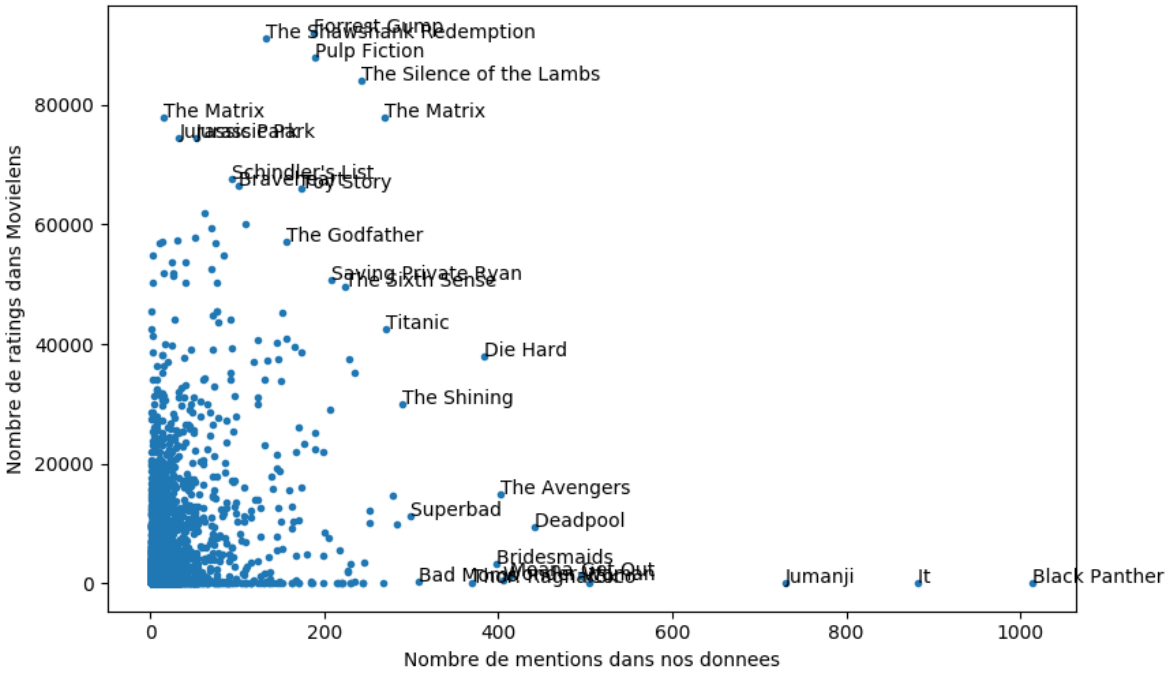
Figure 4.4 Histogramme du nombre de mentions par film. Les deux axes sont en échelle logarithmique.

une fois dans notre jeu de données. Les coordonnées des points sont respectivement le nombre de mentions du film dans notre jeu de données, et le nombre de *ratings* dans Movielens. On voit sur la figure 4.5b que ces deux quantités sont corrélées positivement. Un certain nombre de films souvent mentionnés dans notre jeu de données n'ont toutefois pas de *rating* dans Movielens (ces films sont affichés avec 1 *rating* par soucis de présentation). En effet, certains films n'ont pas pu être appareillés dans Movielens, comme expliqué en section 4.1.1. D'autres n'étaient tous simplement pas sortis lors de la publication de la dernière version de Movielens, en août 2017. Le figure 4.5a met en évidence les films les plus populaires. Les films "Jumanji", "It" et "Black Panther" rassemblent un nombre considérable de mentions dans nos données. Ces films à gros budget sont en effet sortis pendant, ou peu de temps avant la récolte des données. En plus de ces trois cas précis, ce phénomène s'observe sur tout le jeu de données : les films récents ont tendance à être plus fréquemment mentionnés. Ce constat laisse penser que la durée entre la sortie du film et le dialogue pourrait être un paramètre à prendre en compte dans des travaux futurs.

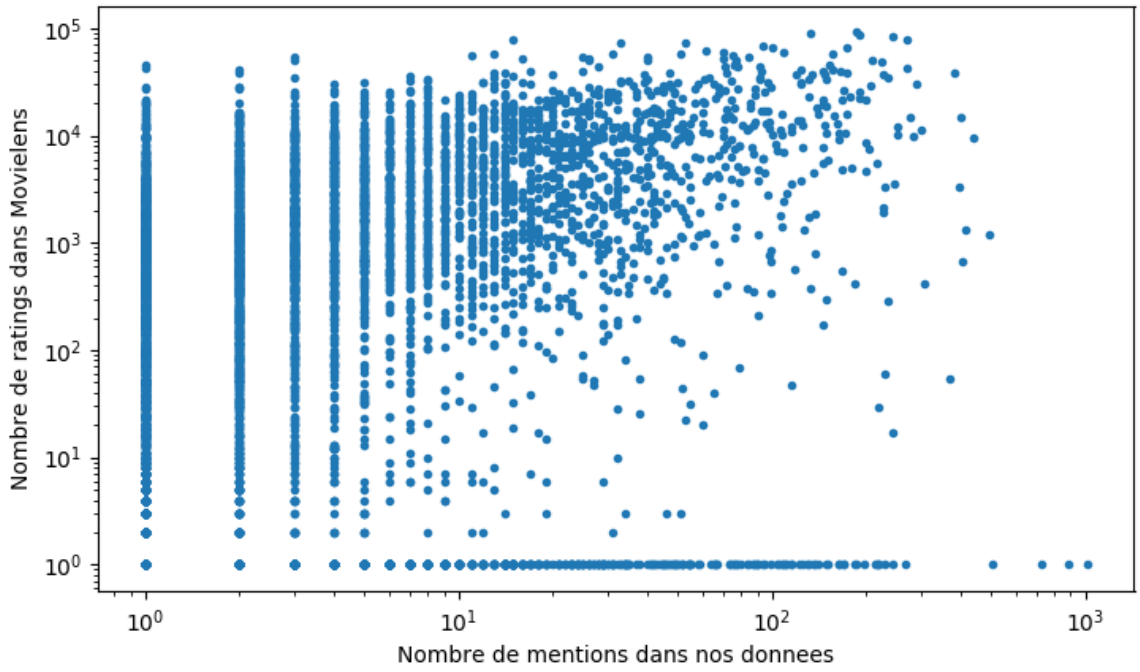
En tout, 963 personnes différentes ont participé à la récolte de données. La figure 4.6 montre l'histogramme du nombre de conversations approuvées par participant. Quasiment la moitié des participants, 433, n'ont qu'une seule conversation. 104 travailleurs ont atteint leur limite en complétant plus de 100 conversations. Si notre tâche n'a pas plu à tous les participants, on constate qu'un nombre conséquent a toutefois bien aimé contribuer à la collection de données. La figure 4.7 est une représentation graphique des participants basée sur le nombre de fois où deux participants ont dialogué ensemble. Seule la composante connexe principale est représentée. Le graphe complet comporte également 15 composantes de deux participants, et une composante de 5 participants. Nous voulions que le jeu de test n'ait pas de participant en commun avec le jeu d'entraînement. Pour extraire un jeu de test du jeu de données initialement collecté et représenté ici, il aurait fallu couper ce graphe en deux partitions, et donc supprimer toutes les arrêtes de coupe. La densité du graphe en figure 4.7 montre qu'on aurait perdu un grand nombre de conversations pour faire cette coupe, et justifie la collecte d'un jeu de test à part entière.

Enfin, la table 4.1 donne le nombre de fois que chaque genre de film a été mentionné dans le jeu de données. La liste de genre correspond à celle donnée par le site IMDb [3]. En tout, 7434 conversations sur les 10021 conversations comprennent au moins une mention de genre. Ces données pourraient donc servir à développer d'autres approches utilisant les genres des films, ou des méta-données de films de manière générale.

3. `https://www.imdb.com/feature/genre/`



(a) Les films les plus populaires sont labélisés.



(b) Les deux axes sont en échelle logarithmique. On attribue une ordonnée de 1 aux films qui n'ont pas de *rating* dans Movielens pour pouvoir les afficher sur l'échelle logarithmique. Ces films forment la ligne de points en bas du graphe.

Figure 4.5 Nuage de points des films : nombre de mentions dans notre jeu de données en abscisse, nombre de *ratings* dans Movielens en ordonnée.

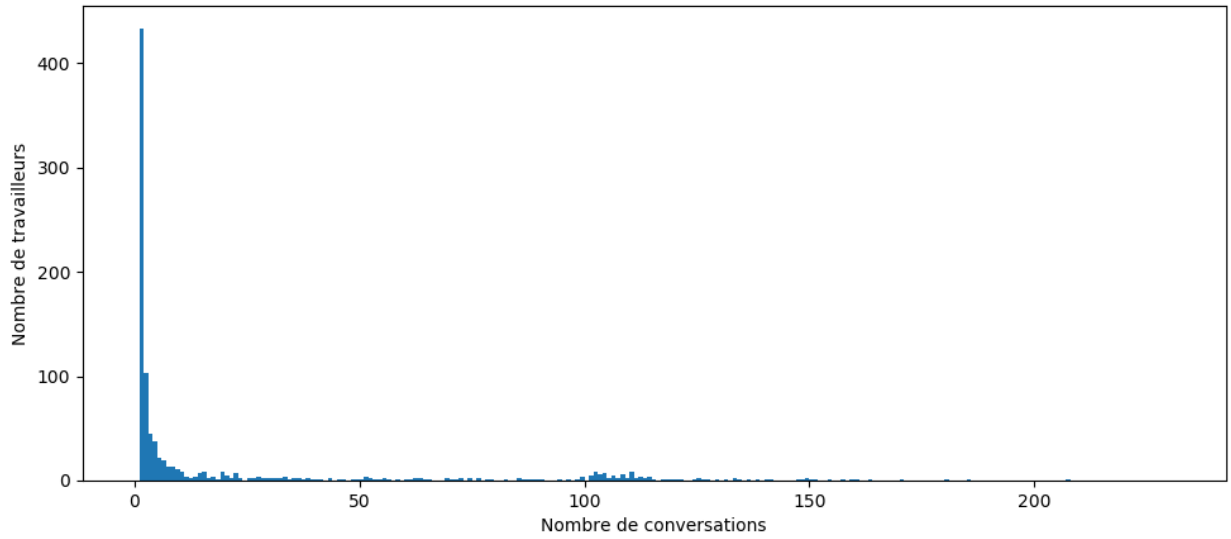


Figure 4.6 Histogramme du nombre de conversations par travailleur.

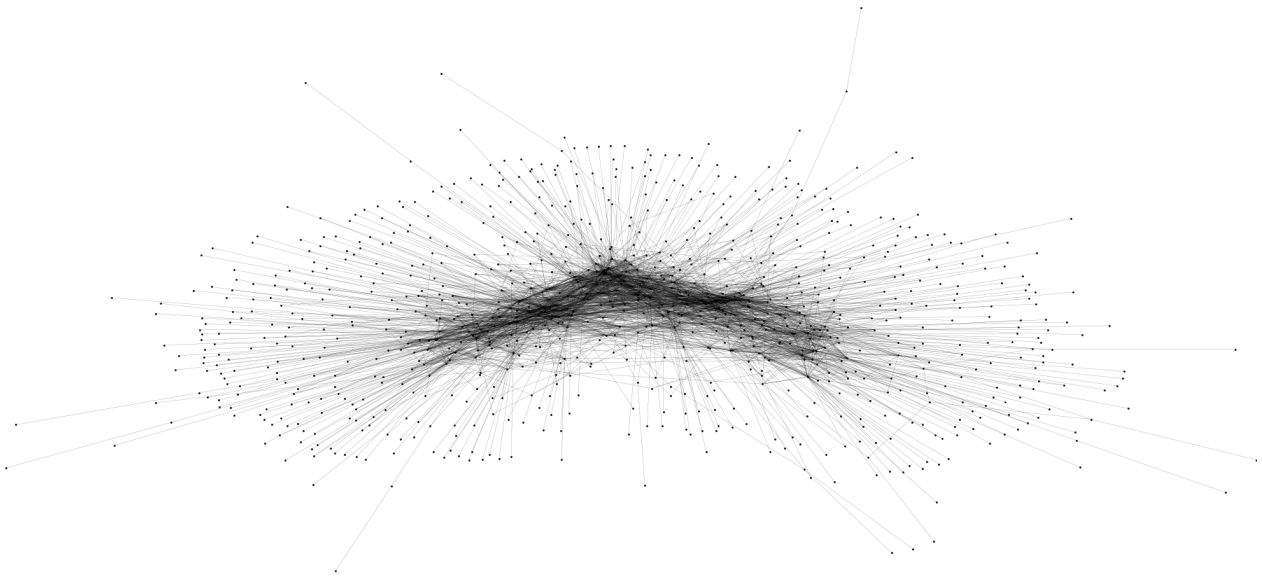


Figure 4.7 Composante connexe principale du graphe des participants. Deux participants sont reliés par une arrête s'ils ont dialogué au moins une fois ensemble. Chaque arrête est pondérée par le nombre de fois où les deux participants ont dialogué ensemble.

4.1.3 Génération de phrases

Comme expliqué en section 3.6, les générations de notre modèle sont obtenues avec une recherche en faisceaux. Nous revenons ici plus en détails sur l'algorithme de génération

Tableau 4.1 Nombre de mentions pour chaque genre. Les données sont tokenisées, puis on compte le nombre de fois où chaque genre, ou son pluriel, apparaît. En plus de “sci-fi”, nous cherchons également les occurrences de “science fiction”. Pour “film noir” et “science fiction”, nous avons simplement compté le nombre de fois où la sous-chaîne de caractère apparaît dans le jeu de données.

action	2922	family	881	romance	498
adventure	249	fantasy	358	sci-fi	415
animation	149	film noir	8	short	23
biography	5	history	64	sport	16
comedy	4363	horror	1880	superhero	242
crime	164	music	223	thriller	907
documentary	241	musical	333	war	313
drama	1283	mystery	275	western	168

de notre agent de dialogue. Actuellement, les recommandations faites par notre modèle sont entièrement décidées par le module de recommandation Autorec (Sedhain et al., 2015). Rappelons que ce dernier est conditionné par les films déjà mentionnés dans la conversation, et produit un vecteur de recommandation $\hat{\mathbf{r}} \in \mathbb{R}^{|V'|}$. Ainsi, le modèle ne fait pas la distinction entre les films déjà mentionnés et les autres. Avec une recherche en faisceau standard, notre agent recommande systématiquement un film qui a déjà été mentionné et que la personne affirme avoir aimé. En effet, quoi de plus certain que la personne aimera un film, lorsque cette personne vient d’affirmer aimer ce film. Pour éviter ce genre de comportement, nous forçons notre modèle à ne recommander que des films pas encore mentionnés dans la conversation.

Ainsi, dès qu’un film a été mentionné, ou généré au cours d’une conversation, nous l’enlevons de la liste des films pour la suite du dialogue. Par ailleurs, pour augmenter la diversité des films recommandés, nous échantillonnons un film avant chaque génération de message (dans le cas contraire, la recherche en faisceaux fait que ce sont en général les films les plus populaires qui apparaissent).

Ce comportement forcé n’est pas une solution viable au problème des recommandations conversationnelles. En effet, il peut arriver que le recommandeur veuille revenir sur des films déjà mentionnés, par exemple “Pourquoi avez-vous bien aimé le film XXX”. Il s’agit plutôt d’un accommodement nous permettant de montrer ce qu’un simple modèle est capable d’apprendre grâce au jeu de données récolté.

4.1.4 Continuation de la récolte de données

Le jeu de données collecté comprend environ 10000 conversations, ce qui nous a permis d'effectuer quelques premières expériences. Toutefois les jeux de données de dialogue utilisés en apprentissage profond sont en général encore plus grands : le corpus de dialogue Ubuntu (Lowe et al., 2015) contient près d'un million de dialogues, le Facebook Movie Dataset (Dodge et al., 2015) en contient environ 3.5 millions. Afin de déterminer si récolter plus de données serait utile, nous avons exécuté quelques expériences. Tout en gardant le même jeu de validation, nous augmentons progressivement la taille du jeu d'entraînement et observons à quel point cette augmentation produit des gains de performance. Nous analysons la perte de validation de notre système complet de dialogue de recommandation (section 3.4.4). La figure 4.8 suggère que la perte diminuerait encore avec plus de données. Il pourrait être pertinent de continuer la récolte de données afin d'obtenir un modèle plus performant.

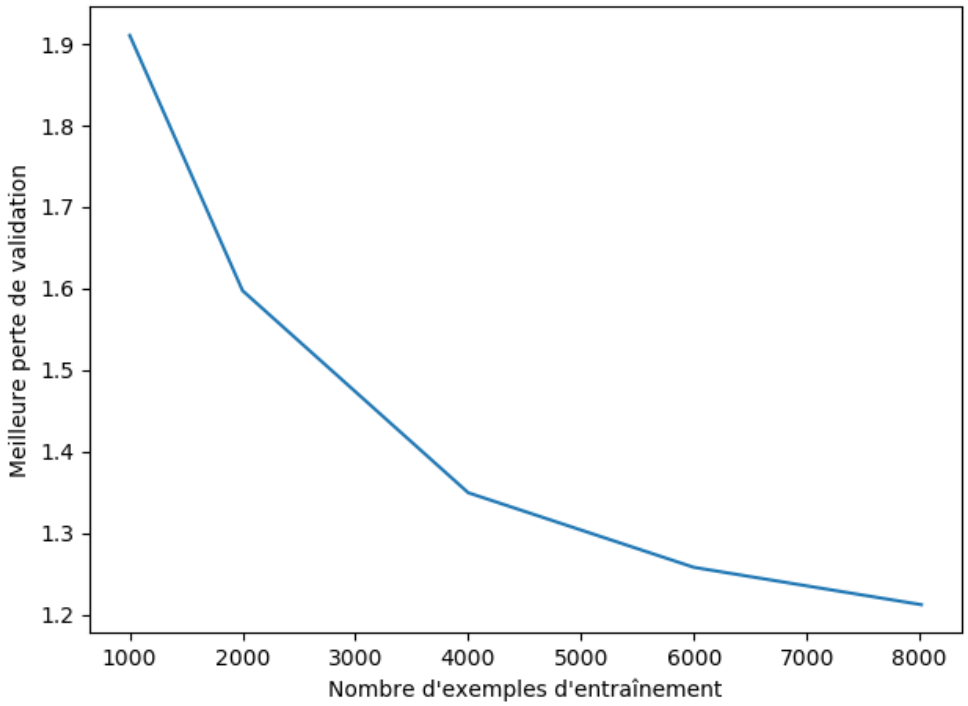


Figure 4.8 Meilleure perte de validation du système de dialogue de recommandation en fonction du nombre d'exemples d'entraînement.

4.2 Autre approches explorées

4.2.1 *Variational Hierarchical Recurrent Encoder-Decoder*

Notre agent de dialogue comporte plusieurs défauts. Tout d'abord, il a le défaut de générer en grande partie des phrases très génériques. Ce problème est bien connu des réseaux de neurones génératifs. Une approche possible pour pallier ce problème est d'introduire de la stochasticité dans la génération. Le Variational Hierarchical Recurrent Encoder-Decoder (VHRED) est une variation du modèle Hierarchical Recurrent Encoder-Decoder (HRED) où Serban et al. (2017a) introduit une variable latente $\mathbf{z}_m \in \mathbb{R}^{d_z}$ à chaque message. La figure 4.9 montre l'architecture du modèle. La génération de phrase se fait donc en deux étapes : on tire d'abord un échantillon de la variable latente, puis on génère la phrase, conditionnellement à la variable latente. La distribution de cette variable est conditionnée sur le contexte de la conversation et est entraînée en maximisant une borne inférieure de la log-vraisemblance, inspirée des travaux sur les auto-encodeurs variationnels (Kingma and Welling, 2013). En reprenant les notations établies en section 3.4.1, cette borne s'écrit :

$$\begin{aligned}\log P(U_1, ..., U_M) \geq \sum_{m=1}^{M} &- \mathrm{KL}[Q(\mathbf{z}_m|U_1, ..., U_m)\|P(\mathbf{z}_m|U_1, ..., U_{m-1})] \\ &+ \mathbb{E}_{Q(\mathbf{z}_m|U_1,...,U_m)}\left[\log P(U_m|\mathbf{z}_m, U_1, ..., U_{m-1})\right]\end{aligned} \tag{4.1}$$

pù Q est l'approximation de la vraie distribution *a posteriori*. $KL[Q\|P]$ est la divergence de Kullback-Leibler de P par rapport à Q. Intuitivement, la paramétrisation *a posteriori* de Q est entraînée simplement par maximisation de la vraisemblance (le deuxième terme de la somme de l'équation 4.1). La minimisation du terme de divergence de Kullback-Leibler à tendance à rapprocher $P(\mathbf{z}_m|(U_1, ..., U_{m-1})$ et $Q(\mathbf{z}_m|U_1, ..., U_m)$. Au moment de la génération, c'est la distribution P qui est utilisée pour échantillonner les variables $\mathbf{z}_m$. Cette méthode donne plus de variabilité aux générations du modèle. De plus, la variable latente permet de représenter l'information haut-niveau de la phrase de sortie, tandis que l'état caché du réseau de décodage peut se concentrer sur l'information plus bas-niveau.

Ce modèle n'a pas mené à des générations particulièrement plus diversifiées malheureusement. Nous avons également mesuré l'entropie de la distribution des mots générés :

$$H_w = -\sum_{w \in V} p_w \log(p_w) \tag{4.2}$$

où p_w représente la fréquence d'apparition du mot w dans les générations du modèle sur l'ensemble de validation. L'entropie donne une mesure de l'information, ou de la diversité

présente dans la distribution considérée. Le tableau 4.2 montre que cette modification n'apporte pas plus de diversité aux générations. Les résultats de Serban et al. (2017a) sont toutefois basés sur un jeu de données de plusieurs millions de dialogues. Il est raisonnable de penser que la taille restreinte de notre jeu de données empêche le modèle VHRED de générer des phrases plus diversifiées.

Tableau 4.2 Entropie des générations des différents modèles.

Modèle	Entropie H_w des générations
basé sur HRED	4.15
basé sur VHRED	4.07
humain	4.91

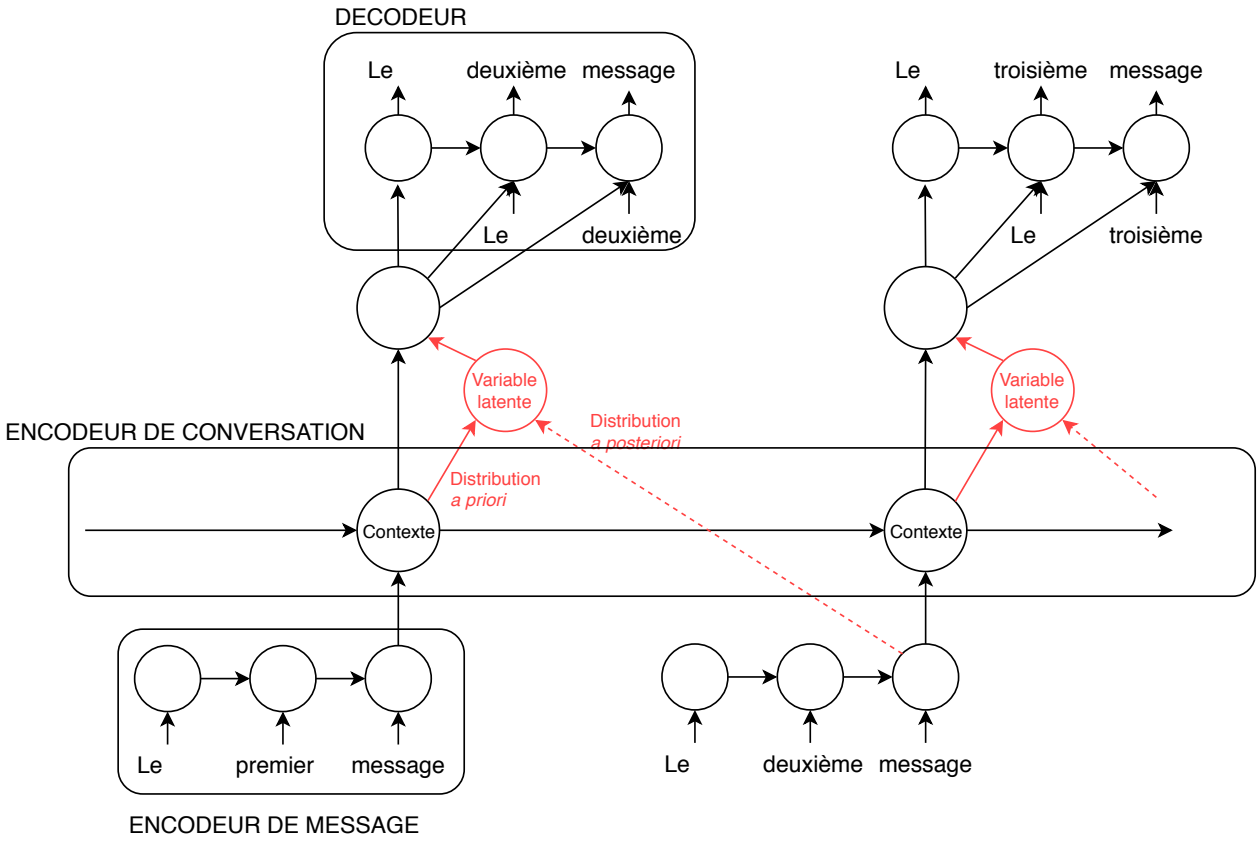


Figure 4.9 Architecture du VHRED. Par rapport au HRED, le VHRED ajoute les parties en rouge.

4.2.2 Système de recommandation conditionné sur le langage

Notre système de recommandation est uniquement basé sur les films qui ont été mentionnés dans la conversation. Il s'agit d'un autre défaut important de notre système, puisqu'il ne pourrait pas proposer de recommandation pertinente à quelqu'un qui demanderait simplement "Je voudrais voir un bon film de science-fiction". Ce genre de requête arrive souvent dans notre jeu de données, et c'est tout l'intérêt des recommandations conversationnelles : l'utilisateur peut préciser par le langage quel genre de recommandation il cherche. Ainsi, nous modifions légèrement notre architecture pour que les recommandations prennent également en compte le langage, en plus des films mentionnés par l'utilisateur. Nous proposons pour cela d'introduire une connexion entre la contexte de la conversation et la dernière couche cachée de l'auto-encodeur de recommandation, qui sert de représentation d'utilisateur. Ainsi, au fil de l'entraînement, le modèle apprendrait à affiner le profil de l'utilisateur à partir des messages reçus. Cette connexion est représentée par la flèche rouge dans la figure 4.10.

La connexion supplémentaire ajoute des paramètres au modèle. Dans les quelques expériences menées, cette modification a malheureusement toujours amené le modèle au sur-apprentissage, par rapport au modèle sans conditionnement sur le langage. À savoir que la meilleure perte de validation est plus forte que celle de notre modèle basique. Nous avons réduit la taille de la représentation de l'utilisateur à 30 dimensions pour réduire ce sur-apprentissage, mais sans succès. On peut supposer que ce phénomène disparaîtra avec un jeu de données plus important, et que cette modification permettrait de faire des recommandations prenant effectivement en compte le genre demandé par l'utilisateur, ou d'autres critères.

4.2.3 Pré-entraînement du modèle sur le jeu de données de Reddit

Nous avons également examiné un modèle pré-entraîné sur le jeu de conversations Reddit sur les films (Dodge et al., 2015). Le même modèle décrit en section 3.4 est tout d'abord entraîné en utilisant le jeu de données Reddit. Puis, les poids du modèle sont affinés sur notre jeu de données, de plus petite taille. L'avantage de ce genre de pratiques est que le modèle peut acquérir des connaissances générales lors du pré-entraînement, qui faciliteront l'entraînement par la suite. Par exemple, l'agent de dialogue pourrait apprendre à comprendre et à modéliser le langage spécifique aux films sur ce large jeu de données. Ce pré-entraînement permet en effet d'obtenir une meilleure vraisemblance sur l'ensemble de validation. On constate cependant un défaut principal dans les générations de phrases : le modèle ne recommande quasiment pas de film. En effet, le jeu de données Reddit ne comprend pas de mentions exactes de films comme notre jeu de données. Ainsi le modèle pré-entraîné sur ce jeu de données apprend exclusivement la partie linguistique. Notamment, le *switch* du décodeur explicité en partie

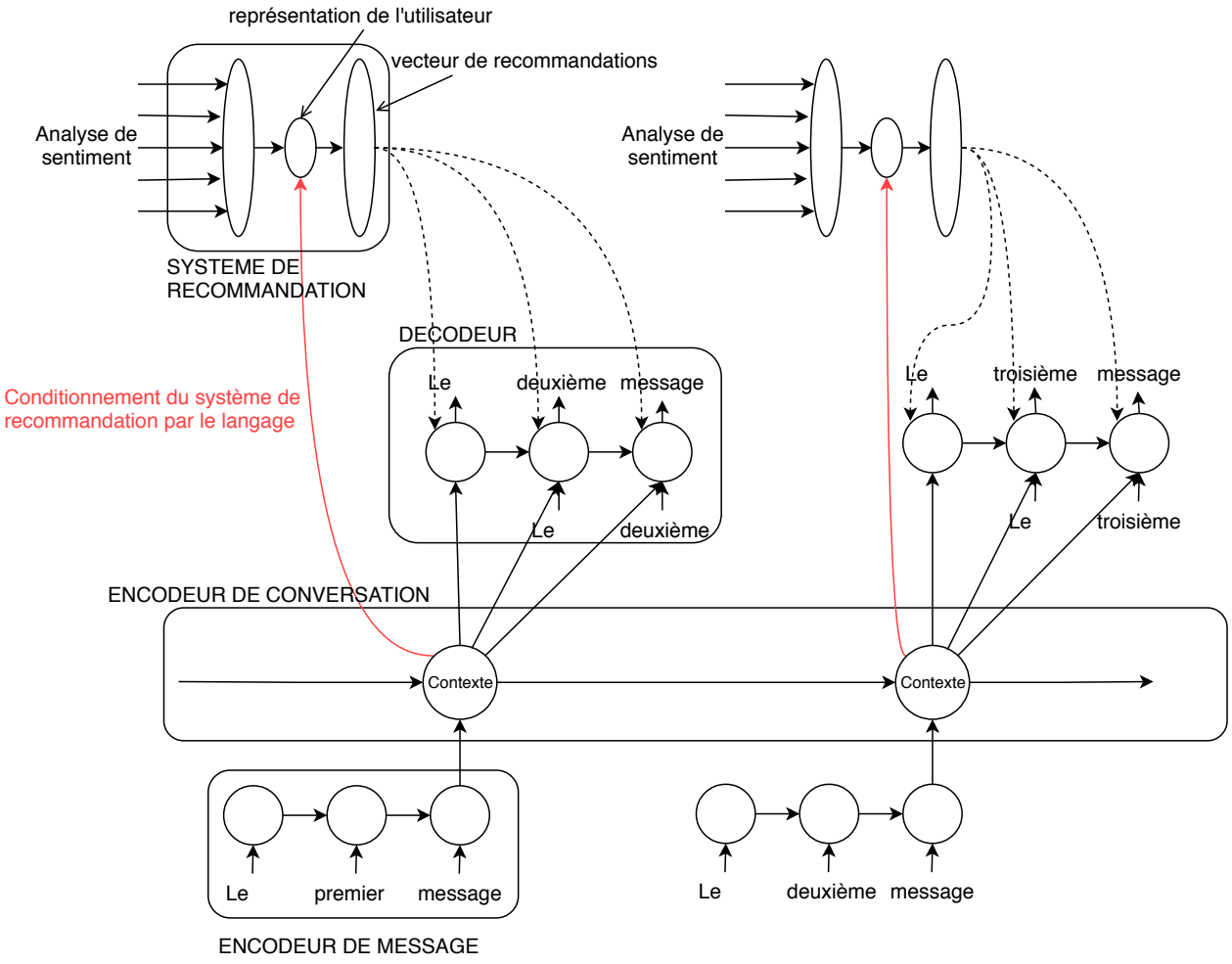


Figure 4.10 Architecture modifiée pour prendre en compte le langage dans les recommandations. Les flèches en pointillées signifient que l'état caché du décodeur n'est pas conditionné sur le vecteur de recommandation, ce dernier est seulement utilisé pour produire l'output du modèle comme décrit en section 3.4.4.

3.4.4 apprend à toujours prédire un *token* dans l'ensemble V, et jamais dans l'ensemble V'. Le modèle est donc devenu très bon dans la compréhension du langage, mais ne répond toutefois pas à la tâche de recommandations conversationnelles. Il serait intéressant de retrouver les mentions de films (ou en partie au moins) dans le jeu de données Reddit pour rendre ce jeu de données plus semblable au notre.

Cette expérience révèle un problème assez fondamental de la fonction de perte que nous utilisons dans notre approche. Un modèle parvient à atteindre une perte sensiblement plus faible, mais toutefois répond moins à nos attentes dans les faits. Il serait donc intéressant d'entraîner des modèles avec des fonctions de perte différentes. Par exemple, nous pourrions donner plus d'importance aux recommandations, ou au fait de choisir correctement entre les deux dictionnaires V et V'.

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

Dans le but de développer un système de recommandations conversationnelles, nous avons récolté un nouveau jeu de données dédié à cette tâche. La récolte s'est faite par le biais de la plate forme Amazon Mechanical Turk. Ce jeu de données de plus de 10000 conversations est unique en son genre et permettra de nombreuses avancées dans les systèmes de dialogue et de recommandation lorsqu'il sera rendu public. En effet, les réseaux de neurones promettent de produire des systèmes de dialogues de plus en plus réalistes. Ces données serviront de terrain d'expérimentation pour développer de nouveaux chatbots de recommandation.

Nous avons entraîné un modèle de réseaux de neurones basé sur un *Hierarchical Recurrent Encoder-Decoder* (Sordoni et al., 2015) avec notre jeu de données. Notre nouvelle approche propose d'inclure explicitement un système de recommandation dans le modèle de dialogue. Un module d'analyse de sentiment détermine si l'utilisateur a aimé les films mentionnés dans la conversation, puis le système de recommandation utilise cette information pour prédire des recommandations. Enfin, notre décodeur utilise un aiguillage pour inclure ces recommandations dans la génération de texte. De cette manière, le système de recommandation peut être pré-entraîné, tirant ainsi profit de connaissances additionnelles. Nous utilisons également les représentations de langage générales de Subramanian et al. (2018) pour compenser la taille limitée du jeu de données. Notre approche démontre comment combiner de nombreuses sources de données pour entraîner un modèle raisonnable. Chaque sous-composante de notre modèle (l'analyse de sentiment et le système de recommandation) est évaluée indépendamment, et nous évaluons le modèle complet avec une étude humaine. Cette étude montre que notre modèle a su tirer profit des autres sources de données utilisées, et donne de meilleurs résultats qu'un *Hierarchical Recurrent Encoder-Decoder* entraîné sur les 10000 conversations.

Pour augmenter la diversité des générations du modèle, nous examinons le *Variational Hierarchical Recurrent Encoder-Decoder*, qui ajoute de la stochasticité aux générations. Nous tentons également de conditionner les recommandations sur le langage. Si ces deux approches n'ont pas donné de résultat convaincant, elles laissent penser qu'on gagnerait à récolter davantage de données.

5.2 Limitations de la solution proposée et améliorations futures

Notre approche présente plusieurs limitations. La première est la taille restreinte du jeu de données collecté. Le jeu de données était suffisant pour entraîner un modèle basé sur des architectures standards de dialogue. Toutefois les jeux de données de dialogue sont en général beaucoup plus grands, certains ayant plus de 100 fois plus de conversations. Cette limitation s'est fait sentir lorsque nous avons voulu complexifier le modèle, ou augmenter la taille et le nombre de couches cachées. Nous pourrions ainsi continuer la récolte des données. Il est probable que récolter deux ou trois fois plus de données suffise à produire de meilleurs résultats.

Notre première approche produisait des recommandations qui ne prenaient en compte que les films mentionnés dans la conversation, mais pas toute la conversation en elle même. Par exemple, des requêtes de genre de film spécifique ne seraient pas prises en compte. Nous avons proposé d'ajouter une connexion entre le contexte du dialogue et la représentation d'utilisateur utilisée dans le système de recommandation. Cette solution n'a malheureusement pas donné de résultat convaincant, mais il est possible qu'elle s'améliore avec plus de données. Notre approche souffre par ailleurs d'un problème bien connu des réseaux de neurones appliqués au langage, qui est le manque de diversité des phrases générées. Le modèle a tendance à trop privilégier les phrases peu informatives et génériques comme "Je ne sais pas".

Le modèle proposé est aussi incohérent dans le sens où le vecteur de recommandation ne conditionne pas les mots utilisés dans la génération de phrase. A contrario, lorsqu'un humain recommande un film, il sait déjà au début de la phrase quel film il va recommander. Par exemple dans la phrase "Comme bon film d'action, je peux te proposer Mad Max", le début est déjà conditionné par le fait que la personne va recommander le film Mad Max. Il serait intéressant de choisir dès le début de la génération le film qui sera mentionné (ou non), puis de conditionner la phrase sur ce choix de film.

L'expérience où le modèle est pré-entraîné sur le jeu de données Reddit a fait observer que la fonction de coût utilisée pourrait ne pas être entièrement adaptée à notre problème. Nous pourrions modifier la fonction de coût pour accorder plus d'importance aux recommandations de films, ou au fait de pondérer entre les dictionnaires V et V'.

Enfin, l'évaluation humaine proposée en section 3.5 est un peu rudimentaire. On demande aux participants d'évaluer la "pertinence" des réponses, ce qui peut sembler vague et assez subjectif. Il serait intéressant de réaliser cette évaluation à nouveau en posant des questions plus précises aux participants, comme par exemple : Quelle est la réponse qui donne la meilleure recommandation de film ? Quelle est la réponse qui a le meilleur niveau de langage ?

Quelle est la réponse que vous auriez donnée ? Ce genre d'évaluation nous donnerait une meilleure idée des modules à améliorer, et montrerait à quelle point le modèle a su combiner dialogue et recommandations.

De nombreuses autres approches à ce problème sont possibles, et nous donnons dans ce dernier paragraphe des pistes qui seraient intéressantes à explorer. Nous pourrions utiliser des données de dialogue synthétiquement générées à partir de Movielens, comme le jeu de recommandation du Facebook Movie Dialog Dataset, pour pré-entraîner le modèle. Il ne serait plus nécessaire d'intégrer explicitement un système de recommandation. Une autre direction de recherche serait d'utiliser un graphe de connaissance sur le domaine des films pour créer un modèle plus savant. On remarque en effet que pour donner des recommandations adéquates il faut parfois avoir une connaissance des méta-données des films (les acteurs, le réalisateur, le genre, etc.). Il serait intéressant de développer un modèle capable d'utiliser ces connaissances pour être capable conjointement de faire des recommandations, et de répondre à des questions factuelles.

RÉFÉRENCES

A. Almahairi, K. Kastner, K. Cho, et A. Courville, “Learning distributed representations from reviews for collaborative filtering”, dans *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 147–154.

H. Aust, M. Oerder, F. Seide, et V. Steinbiss, “The philips automatic train timetable information system”, *Speech Communication*, vol. 17, no. 3-4, pp. 249–262, 1995.

D. Bahdanau, K. Cho, et Y. Bengio, “Neural machine translation by jointly learning to align and translate”, *arXiv preprint arXiv :1409.0473*, 2014.

L. Bahl, P. Brown, P. De Souza, et R. Mercer, “Maximum mutual information estimation of hidden markov model parameters for speech recognition”, dans *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’86.*, vol. 11. IEEE, 1986, pp. 49–52.

R. E. Banchs, “Movie-dic : a movie dialogue corpus for research and development”, dans *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 203–207.

T. Bansal, D. Belanger, et A. McCallum, “Ask the gru : Multi-task learning for deep text recommendations”, dans *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 107–114.

S. Bengio, O. Vinyals, N. Jaitly, et N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks”, dans *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

R. Bisiani, “Beam search”, *Encyclopedia of Artficial Intelligence*, pp. 1467–1468, 1992.

D. G. Bobrow, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, et T. Winograd, “Gus, a frame-driven dialog system”, *Artificial intelligence*, vol. 8, no. 2, pp. 155–173, 1977.

P. Bojanowski, E. Grave, A. Joulin, et T. Mikolov, “Enriching word vectors with subword information”, *arXiv preprint arXiv :1607.04606*, 2016.

L. Breiman, “Random forests”, *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

J. Cassell, *Embodied conversational agents*. MIT press, 2000.

H. Chen, X. Liu, D. Yin, et J. Tang, “A survey on dialogue systems : Recent advances and new frontiers”, *arXiv preprint arXiv :1711.01731*, 2017.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, et Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, 2014.

N. Chomsky, “Three models for the description of language”, *IRE Transactions on information theory*, vol. 2, no. 3, pp. 113–124, 1956.

K. Christakopoulou, F. Radlinski, et K. Hofmann, “Towards conversational recommender systems”, dans *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 815–824.

J. Cohen, “A coefficient of agreement for nominal scales”, *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

C. Cortes et V. Vapnik, “Support-vector networks”, *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

G. Cybenko, “Approximation by superpositions of a sigmoidal function”, *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, et D. Batra, “Visual dialog”, dans *Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017.

J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, et J. Weston, “Evaluating prerequisite qualities for learning end-to-end dialog systems”, *arXiv :1511.06931 [cs.CL]*, 2015.

M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, et M. Galley, “A knowledge-grounded neural conversation model”, *arXiv preprint arXiv :1702.01932*, 2017.

M. H. Göker, P. Langley, et C. A. Thompson, “A personalized system for conversational recommendations”, *arXiv :1107.0029 [cs.IR]*, 2011.

P. K. Gopalan, L. Charlin, et D. Blei, “Content-based recommendations with poisson factorization”, dans *Advances in Neural Information Processing Systems*, 2014, pp. 3176–3184.

C. Greco, A. Suglia, P. Basile, et G. Semeraro, “Converse-et-impera : Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems”, dans *AI*IA 2017 Advances in Artificial Intelligence - XVIth International Conference of the Italian Association for Artificial Intelligence*, 2017, pp. 372–386.

C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, et Y. Bengio, “Pointing the unknown words”, *arXiv :1603.08148 [cs.CL]*, 2016.

F. M. Harper et J. A. Konstan, “The movielens datasets : History and context”, *Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.

T. Hastie, R. Tibshirani, et J. Friedman, “Unsupervised learning”, dans *The elements of statistical learning*. Springer, 2009, pp. 485–585.

G. E. Hinton et R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, *science*, vol. 313, no. 5786, pp. 504–507, 2006.

G. E. Hinton, J. L. McClelland, D. E. Rumelhart *et al.*, “Distributed representations”, *Parallel distributed processing : Explorations in the microstructure of cognition*, vol. 1, no. 3, pp. 77–109, 1986.

S. Hochreiter, “Untersuchungen zu dynamischen neuronalen netzen”, *Diploma, Technische Universität München*, vol. 91, p. 1, 1991.

S. Hochreiter et J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

M. Jacomy, T. Venturini, S. Heymann, et M. Bastian, “ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software”, *PloS ONE*, vol. 9, no. 6, 2014.

P. Johansson, “Design and development of recommender dialogue systems”, Thèse de doctorat, Institutionen för datavetenskap, 2004.

D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization”, *arXiv preprint arXiv :1412.6980*, 2014.

D. P. Kingma et M. Welling, “Auto-encoding variational bayes”, *arXiv preprint arXiv :1312.6114*, 2013.

Y. Koren, R. Bell, et C. Volinsky, “Matrix factorization techniques for recommender systems”, *Computer*, vol. 42, no. 8, 2009.

B. Krause, M. Damonte, M. Dobre, D. Duma, J. Fainberg, F. Fancellu, E. Kahembwe, J. Cheng, et B. Webber, “Edina : Building an open domain socialbot with self-dialogues”, *arXiv :1709.09816 [cs.CL]*, 2017.

J. Li, M. Galley, C. Brockett, J. Gao, et B. Dolan, “A diversity-promoting objective function for neural conversation models”, *arXiv preprint arXiv :1510.03055*, 2015.

X. Li et D. Roth, “Learning question classifiers”, dans *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 2002, pp. 1–7.

C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, et J. Pineau, “How not to evaluate your dialogue system : An empirical study of unsupervised evaluation metrics for dialogue response generation”, *arXiv preprint arXiv :1603.08023*, 2016.

R. Lowe, N. Pow, I. Serban, et J. Pineau, “The ubuntu dialogue corpus : A large dataset for research in unstructured multi-turn dialogue systems”, *arXiv preprint arXiv :1506.08909*, 2015.

B. M. Marlin, R. S. Zemel, S. T. Roweis, et M. Slaney, “Collaborative filtering and the missing at random assumption”, dans *UAI*, 2007, pp. 267–275.

T. Mikolov, K. Chen, G. Corrado, et J. Dean, “Efficient estimation of word representations in vector space”, *arXiv preprint arXiv :1301.3781*, 2013.

T. Mikolov, W.-t. Yih, et G. Zweig, “Linguistic regularities in continuous space word representations”, dans *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 2013, pp. 746–751.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, et J. Weston, “Parlai : A dialog research software platform”, *arXiv preprint arXiv :1705.06476*, 2017.

B. Pang et L. Lee, “A sentimental education : Sentiment analysis using subjectivity summarization based on minimum cuts”, dans *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 271.

B. Pang, L. Lee *et al.*, “Opinion mining and sentiment analysis”, *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

J. Pennington, R. Socher, et C. Manning, “Glove : Global vectors for word representation”, dans *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

A. Ritter, C. Cherry, et W. B. Dolan, “Data-driven response generation in social media”, dans *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 583–593.

M. Schuster et K. K. Paliwal, “Bidirectional recurrent neural networks”, *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

S. Sedhain, A. K. Menon, S. Sanner, et L. Xie, “Autorec : Autoencoders meet collaborative filtering”, dans *International Conference on World Wide Web*, 2015, pp. 111–112.

R. Sennrich, B. Haddow, et A. Birch, “Neural machine translation of rare words with subword units”, *arXiv preprint arXiv :1508.07909*, 2015.

I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke *et al.*, “A deep reinforcement learning chatbot”, *arXiv preprint arXiv :1709.02349*, 2017.

I. V. Serban, R. Lowe, P. Henderson, L. Charlin, et J. Pineau, “A survey of available corpora for building data-driven dialogue systems”, *arXiv :1512.05742 [cs.CL]*, 2015.

I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, et J. Pineau, “Building end-to-end dialogue systems using generative hierarchical neural network models.” dans *AAAI*, vol. 16, 2016, pp. 3776–3784.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, et Y. Bengio, “A hierarchical latent variable encoder-decoder model for generating dialogues.” dans *AAAI*, 2017.

A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, et J.-Y. Nie, “A hierarchical recurrent encoder-decoder for generative context-aware query suggestion”, dans *International on Conference on Information and Knowledge Management*, 2015, pp. 553–562.

X. Su et T. M. Khoshgoftaar, “A survey of collaborative filtering techniques”, *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

S. Subramanian, A. Trischler, Y. Bengio, et C. J. Pal, “Learning general purpose distributed sentence representations via large scale multi-task learning”, dans *ICLR*, 2018.

A. Suglia, C. Greco, P. Basile, G. Semeraro, et A. Caputo, “An automatic procedure for generating datasets for conversational recommender systems”, dans *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017.

S. Sukhbaatar, J. Weston, R. Fergus *et al.*, “End-to-end memory networks”, dans *Advances in neural information processing systems*, 2015, pp. 2440–2448.

I. Sutskever, J. Martens, G. Dahl, et G. Hinton, “On the importance of initialization and momentum in deep learning”, dans *International conference on machine learning*, 2013, pp. 1139–1147.

R. S. Sutton et A. G. Barto, *Reinforcement learning : An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

H. P. Truong, P. Parthasarathi, et J. Pineau, “Maca : A modular architecture for conversational agents”, *arXiv preprint arXiv :1705.00673*, 2017.

V. N. Vapnik et A. Y. Chervonenkis, “The uniform convergence of frequencies of the appearance of events to their probabilities”, dans *Doklady Akademii Nauk*, vol. 181, no. 4. Russian Academy of Sciences, 1968, pp. 781–783.

P. Vincent, H. Larochelle, Y. Bengio, et P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders”, dans *International Conference on Machine Learning*, 2008, pp. 1096–1103.

W. Wahlster, *SmartKom : foundations of multimodal dialogue systems*. Springer, 2006, vol. 12.

R. S. Wallace, “The anatomy of alice”, dans *Parsing the Turing Test*. Springer, 2009, pp. 181–210.

C. Wang et D. M. Blei, “Collaborative topic modeling for recommending scientific articles”, dans *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

W. Ward et S. Issar, “Recent improvements in the cmu spoken language understanding system”, dans *Proceedings of the workshop on Human Language Technology*. Association

for Computational Linguistics, 1994, pp. 213–216.

P. Wärnestål, L. Degerstedt, et A. Jönsson, “Emergent conversational recommendations : A dialogue behavior approach”, dans *SIGDIAL Workshop on Discourse and Dialogue*, 2007.

J. Weizenbaum, “Eliza—a computer program for the study of natural language communication between man and machine”, *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

J. E. Weston, “Dialog-based language learning”, dans *Advances in Neural Information Processing Systems*, 2016, pp. 829–837.

D. H. Widyantoro et Z. K. A. Baizal, “A framework of conversational recommender system based on user functional requirements”, dans *International Conference on Information and Communication Technology (ICoICT)*, 2014, pp. 160–165. DOI : 10.1109/ICoICT.2014.6914058

R. J. Williams et D. Zipser, “A learning algorithm for continually running fully recurrent neural networks”, *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

C. Xing, W. Wu, Y. Wu, M. Zhou, Y. Huang, et W.-Y. Ma, “Hierarchical recurrent attention network for response generation”, *arXiv preprint arXiv :1701.07149*, 2017.

S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, et J. Weston, “Personalizing dialogue agents : I have a dog, do you have pets too ?” *arXiv preprint arXiv :1801.07243*, 2018.